
FEASIBILITY OF SELF-DRIVING CAR IN A VIRTUAL ENVIRONMENT

Shubham Vohra*¹, Sujata Gaikwad*²

*¹Student, Dr. Babasaheb Ambedkar Technological University, Department of Computer Science and Engineering, College of Engineering Osmanabad, Osmanabad, Maharashtra, India)

*²Head of Department, Dr. Babasaheb Ambedkar Technological University, Department of Computer Science and Engineering, College of Engineering Osmanabad, Osmanabad, Maharashtra, India

ABSTRACT

The Convolutional Neural Networks technique is especially powerful in pattern recognition work using images because the convolutional layer operation grasps the Two-Dimensional nature of given images. Also, by using the convolutional kernels to scan the entire 2D image, relatively very few numbers of parameters need to be learned as compared to the sum total number of performed operations, thus reducing the computational cost. Our work here includes capturing the Two-Dimensional images and passing those to CNN based model which then can generate the precise steering angles, Thus which can be utilized to Drive a vehicle without human intervention,, The prime motive for this research is to steer clear of the need to acknowledge specific designated features, such as road lane marks, guarding rails, cars or pedestrians, and to neglect having to assemble a group of “if, then, else” rules and regulations, based on monitoring of these features learning and extraction ability. This research describes preliminary approaches and results of this new effort.

Keywords: CNN, 2D, GPU, Steering Angle, Udacity Self-Driving Car Simulator, NVIDIA, Autonomous Vehicle, Self-Driving Vehicle, MSE, Convolutional Neural Networks, ILSVRC, Relu, Adam Optimizer, Max Pooling, Flatten layer, Normalization layer, SGD, RMSprop, MSE, EarlyStopping, LR, Gradient, back propagation, learning rate.

I. INTRODUCTION

The problems arising due to manual driving were like safety issues, high traffic, more fuel consumption, more time consumption, and dependency of human intervention for driving. safety issues might arise as manual driving may face human errors while driving like lack of attentiveness, drunk drivers, The Global status report published on road safety for December 2018, issued by WHO (World Health Organization) [18], states that the sum total number of deaths caused due to road accidents has outstretched up to 1.35 million annually. Damage caused due to road accidents are now play a vital role for deaths of people aged between 5-29 years. High traffic issues affect due to lack of inter-car communications, wastage of fuel as well as time due to high traffic, and manually searching for nearby parking spaces, lack of auto-navigation results in time consuming for navigation. Lack of usage of full potential of the infrastructure due to dependency on human intervention for driving, which results in less productivity and not performing at its best constantly. While Convolutional Neural Networks with feature learning capability have been in market for over twenty-five years, their usage has exploded in each and every field in last few years. In this research, we describe a Convolutional Neural Network that can extend and go beyond pattern recognition tasks. It grasps the whole processing line with highly precise feature learning rate and accuracy required to drive a vehicle resulting into a self-driving or an autonomous vehicle.

Technological advances are tremendously growing in an unmanageable manner. The more an individual invests in fruitful innovative technologies, the more reward it enjoys in the future. These rewards include even from finer infrastructure to augmentation reality and various other sectors. The prime motive of technology today is to leverage the abilities of GPUs and IoT devices to reduce the human intervention in everyday tasks. Machine Learning, Deep Learning, Artificial Intelligence, Augmented reality and Computer Vision being the giant role players in doing so. We have reached to industry 5.0 doorstep wherein we can automate our humble tasks like driving vehicles. This segment of Computer Visions and Deep Learning is popularly known as Autonomous Vehicles or Self-Driving Cars. These are vehicles that recognize their environment, surroundings and move without human intervention. With so many country's proceeding towards to pass regulations and so many giant tech and automobile companies testing their vehicles on the roads, you may have already seen a self-driving vehicle like Tesla and maybe you didn't even recognize them. However, giant Tech companies and advanced country's alike are being very confidential about these on road testing's, since the current public point of view is not entirely positive nor negative.

A roadway filled of Self-Driving vehicles would also may improve the standard of driving. Self-Driving vehicles

could connect and coordinate their speed, direction and movements with each other accordingly, while drastically boost their efficiency and accuracy. The result would be a precise and accelerated commute that allows audience to simultaneously synchronize and socialize, perform vivid activities while on the way. In addition, systems or Processing units cannot become fatigued, diverted, inebriated, drowsy or physically disabled; they always follow instructions, obey rules and drive cautiously resulting in a safer driving environment.

II. METHODOLOGY

Steering Angle: The vehicle heading process [14] is defined as in Figure 1 below, zero degrees is north. Vehicle has positive (+) value in anticlockwise direction and negative (-) heading in clockwise.

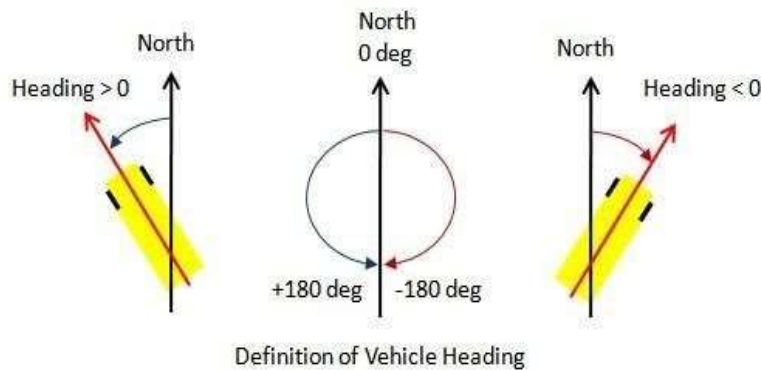


Figure 1: Vehicle Heading in 0 degree, clockwise and anticlockwise [14].

The steering gradient or its angle can be termed as the gradient or angle between the front of the automobile and the automobile wheel axle direction as shown in Figure 2, The automobile steering methodology has a highest steering gradient or angle of

+0.52359878, while the lowest steering gradient is of -0.52359878, radians of highest +30 degree, and lowest of -30 degree.

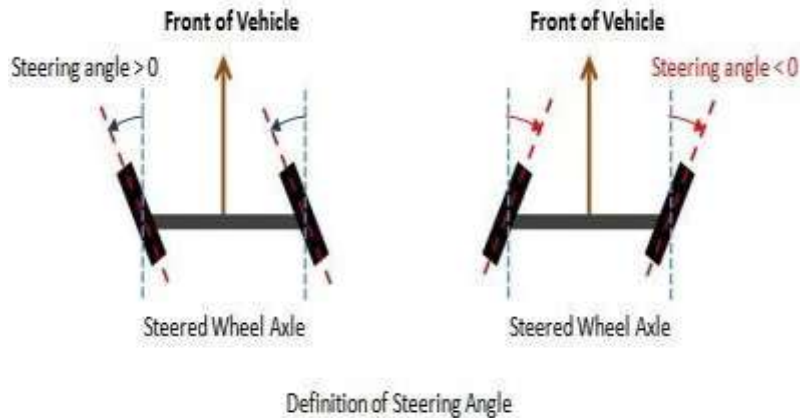


Figure 2: Vehicle Steering Wheel Axle clockwise and anticlockwise [14].

NVIDIA ® had released a paper [1], where they taught Convolutional Neural Networks to synchronize unprocessed pixels (picture element) from an image captured using single front-facing camera mounted on vehicle directly to steering rotation commands. unexpectedly, the outcome was very substantial, as the vehicle learned to drive in actual traffic on roadways with footpath or without footpath patterns or any kind of identification mark on roadways with least quantity of 2D images used as training data. Here, this research uses the simulator [10] provided by udacity®. The simulation car [10] used is provisioned with Three cameras in the front mounted on vehicle itself that records the video and additionally the steering angles corresponding to camera in centre. The captured data thus contains the local system path to left, centre and right images, steering angle generated, throttle, speed and brake floating values.

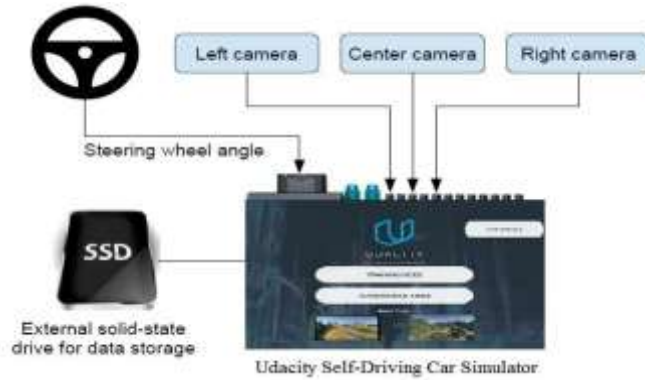


Figure 3: Data Capturing using Udacity Self-Driving Car Simulator [10].

Research includes use of Python 3.8 and Machine Learning as main technologies, Udacity’s Self Driving Car simulator [10] for generating dataset which usually consist of images captured from 3 different camera angles and a .csv file which contains steering angles, speed, throttle and breaking values.



Figure 4: GUI of Udacity’s Self Driving Car simulator [10].

For research, collecting data was a challenge as gathering real-time data would be much costlier as it would have required multiple cameras (3), a computer platform having multiple cores with a GPU chip, an SSD drive for fast read/write operations and data storage & a vehicle for practically driving and recording its real-time data. Therefore, we tried searching for open-sourced available data and came across Udacity’s Self Driving Car simulator [10], Udacity made its self-driving car simulator [10] source code available on their GitHub which was initially developed to educate their Self-Driving Car Nanodegree scholars. Using this Simulator [10], we need to drive a vehicle manually to capture and store training data at required local file system storage, these training data would be stored on a user specified local directory. Training Data would consist of three Images captured from Left, Center & Right for a single frame and Steering Angles captured in a .csv file.



Figure 5: Three Images captured from Left, Center and Right Camera’s.

The .csv file contains the captured data like path to all three images (Left, Center, Right), steering gradient values, vehicle throttling value, vehicle applied break values and even the speed of vehicle. Among these we only require the path to all three images (Left, Center, Right), and steering gradient value.

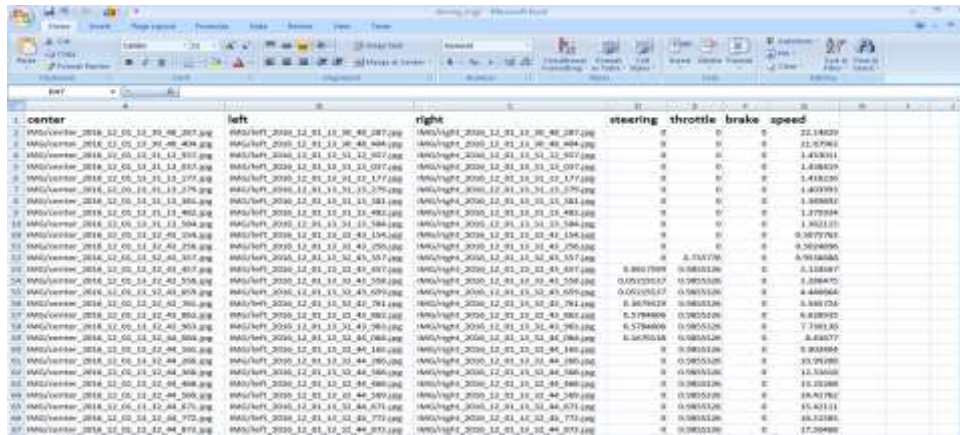


Figure 6: CSV File containing path to images, steering angles, throttle, break and speed values.

For gathering data of different conditions, this simulator [10] has two different lanes for driving the vehicle, we choose the one which had minimal curves and was easy compared to the either one, we tried drive in the center and also to drift the vehicle. This thus provides the training data which could be used to learn features and train the model rectifications.

III. MODELING AND ANALYSIS

While Gathering the training data we collected almost “8036” rows of observations in a .csv file, and captured three times the Image for per observation i.e. (8036 * 3) = “24108” Images in few minutes of driving.

While Analyzing those Actual steering angles captured in a .csv file, found out that most angles lied in between “0.0 – 0.1” as per the driving lane pattern.

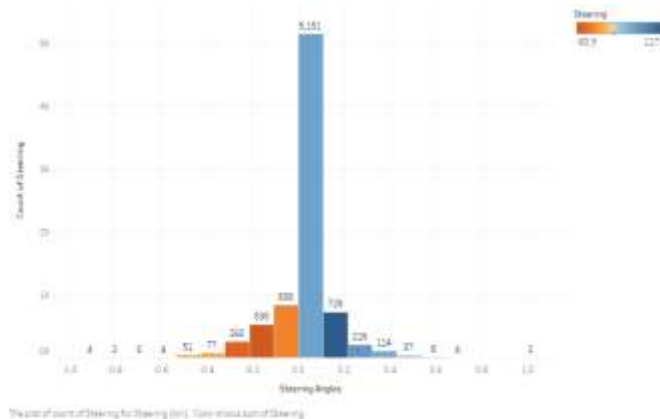


Figure 7: Histogram of Steering Angles (most angles lying between 0.0 – 0.1)

We teach the weights of our neural network to reduce (MSE) mean squared error between the steering command resulting from the network (Predicted Angles) and the angles generated either from the human driver, or the steering angle commands generated by simulator [10] in a .csv file (Actual Angles). The built neural network architecture consisting of multiple layers is demonstrated in Figure 8 below.

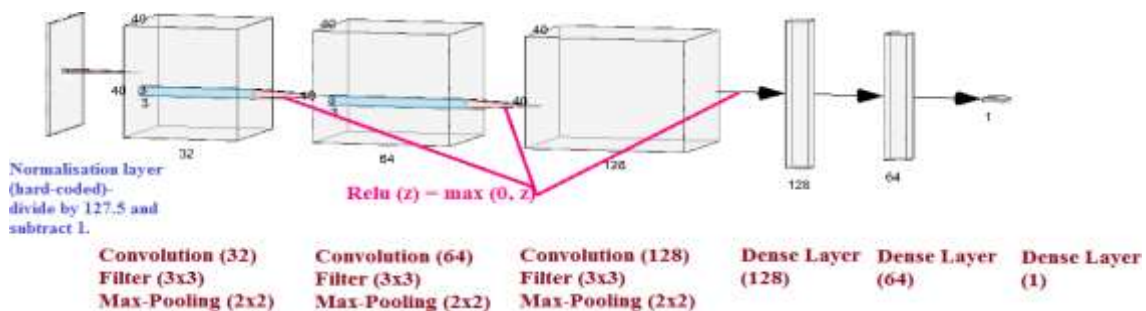


Figure 8: Proposed Neural - Network Architecture.

Our neural network consists of seven layers, including a normalization layer, three connected layers and convolutional layers each. The input unprocessed image is divided into Y, U, V values i.e. YUV planes and passed to the neural network. The first layer of the neural network executes the normalization process on images. The normalization layer has a hard-coded value and it does not fluctuate during the learning process. The key advantage of executing normalization over the input images in the neural network itself permit the updating for normalization scheme with the neural network architecture and to leverage the GPU (Graphics Processing Unit) processing ability.

The Convolutional layers in our neural network model were mapped out in a way that it could execute feature extraction and were selected empirically from a multiple different series of experiments that consists of various different layer configurations. We here adhere to the 3 convolutional layers of size (32, 64, 128) with 3 fully connected layers of size (128, 64, 1) resulting to a control steering value or gradient which is the radius of opposite turning. These 3 fully connected layers are used to mimic as a human controller for vehicle steering

The model has following layers:

- Normalization layer (hard-coded) dividing the value by 127.5 and then out of the resulting value subtract 1, $((\text{value} / 127.5) - 1)$.
- Three convolutional layers of 3*3 kernel with size (32, 64, 128) filters.
- Performing Max-pooling with all above layers.
- A flatten layer.
- In order to perform regularization, after Flatten layer we then perform dropout.
- 128, 64 and 1 output sized three fully connected layers.
- Finally, in order to get steering gradient, the model has a final steering value output layer.

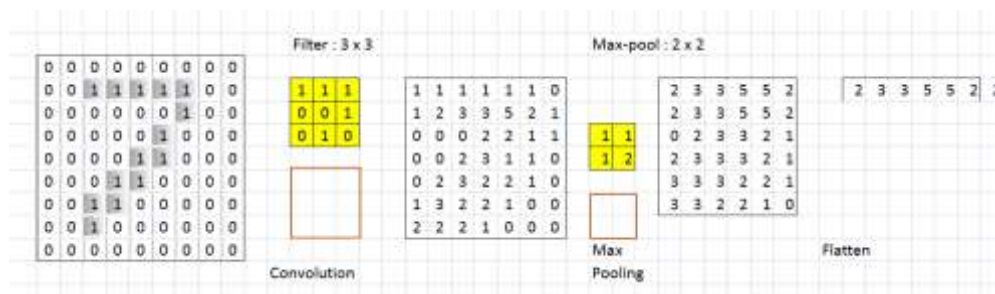


Figure 9: Convolution Filter, Max-Pooling and Flatten Layers Operations.

Raw unprocessed images are fed into a Convolutional Neural Network, which then calculates the predicted steering gradient or angles. The Predicted steering angle is being compared with the expected steering gradient or angle command for the given raw image, also the weights of the Neural Network are adjusted accordingly to bring the actual result closer to the expected result.

The weights updating is carried out using back propagation adjustment as demonstrated in the Image Figure 10 below

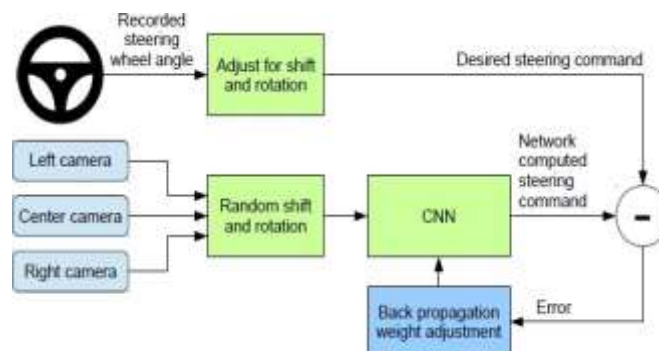


Figure 10: Training Model Architecture [1].

for research we used Adam Optimizer [17] (adaptive moment estimation) with Learning rate of “0.0001” and as Loss function we used MSE (mean squared error).

Adam [17] is dissimilar to the classical (SGD) stochastic gradient descent. SGD keep up a solitary learning rate known as alpha for all the weight renovation and the learning rate won't fluctuate during model training. Adam [17] can be termed as a blend of RMSprop and SGD addition to a force or momentum. It utilizes the gradients (SGD) to scale the learning rate as RMSprop and it leverages the force or momentum by using rolling average of the squared gradient like stochastic gradient descent (SGD) with momentum instead of just gradient alone itself. Backpropagation thus splits these inaccuracies or the difference between actual value and expected value, in backward direction into the neural network model to be used by neurons for renovating each and every weight.

The learning rate (LR) is a hyper-parameter that has a command over the fluctuation of model in response to the calculated error every time the weights in model are changed. selecting the (LR) learning rate is always a tedious task as a learning rate value too minor may result in a higher training time that can get blocked, while a value too big might result in grasping a substandard set of weight too quickly or an unsteady training process. Basically, the learning rate (LR) is a manually configurable parameter used for training the neural networks models that has a very small positive value, always in between the range of 0.0 - 1.0.

The learning rate decides how efficiently the model is grasping the issue. minor learning rates needs high number of model training epochs given the minor updates being made to the weights in each epoch, whereas higher learning rates tends to drastic changes and thus result in low number of model training epochs. A learning rate which is too high can seed the model to rapidly meet to a suboptimal result, while a learning rate (LR) value that is too minor could result in process flow to get blocked and may be tedious or high time consuming. The challenge of teaching deep neural network models involves precise selection of learning rate. As It is one of the principal hyperparameter for model training.

For loss function we selected Mean Squared Error (MSE) and Adam [17] (adaptive moment estimation) optimizer with initial learning rate (LR) of 1e-4 for model training. we also used Early-Stopping call-back on validation loss and carried out up to 5 epochs. we targeted it for 40 epochs for training, but due to Early-Stopping it stopped early at 36th epoch, Figure 11 below shows the reduction of MSE per epoch.

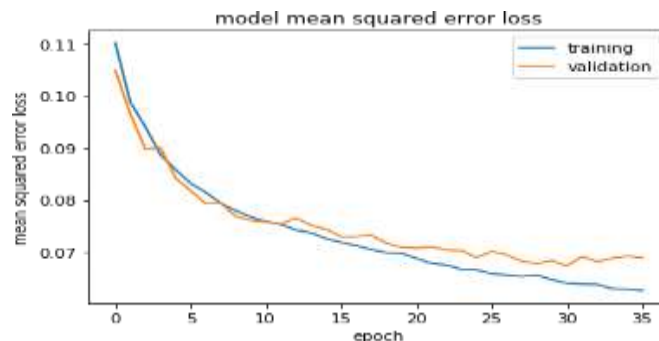


Figure 11: Mean Squared Error (MSE) Reducing Per Epoch.

IV. RESULTS AND DISCUSSION

The Udacity developed simulator [10] takes videos captured from an onboard camera that is forward-facing on a vehicle that is being driven by human and captures the images that mimic the human driving environment that would happen in Convolutional neural network instead of steering. The testing videos are time tuned with generated steering angle values generated by a normal human driver. The simulator [10] ingress the recorded testing video alongside with the tuned steering angle values that were generated when the video was being captured through front facing camera. The simulator [10] application dispatches the initial frames of the selected testing video, as a raw input of the pre-trained CNN model.

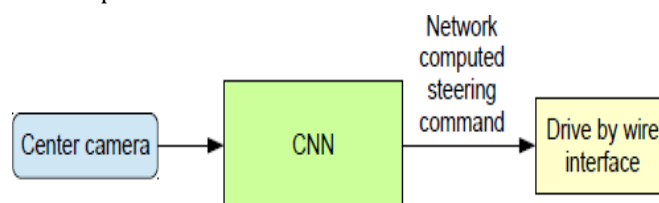


Figure 12: The Pre-trained neural network is used to capture steering angle values from a solitary center camera that is forward-facing [1].

The Convolutional Neural Network model then outputs a steering angle or command for each frame passed. The CNN outputted steering angles and the pre-recorded human-driver angle or commands are then passed into the dynamic CNN model of the automobile to upgrade the orientation and position of the simulated automobile. The Udacity simulator [10] then upgrades the upcoming frame in the testing video to mimic as image would appear as if the car steering should be at position that was outputted by following steering angles from the model. This upcoming image is then passed to the Neural network model and the process iterates.

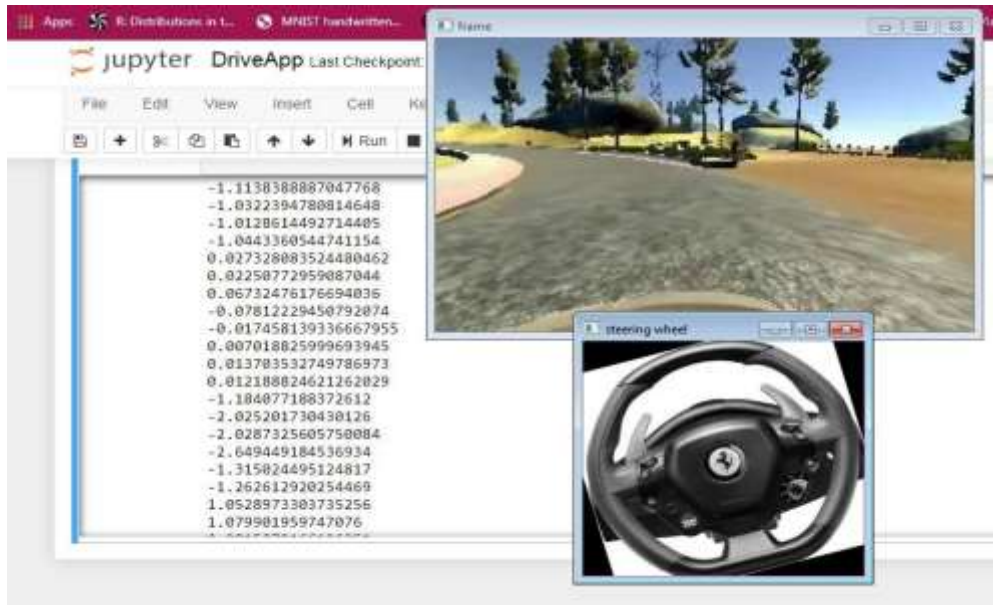


Figure 13: Final output GUI using open-cv's HighGUI module.

V. CONCLUSIONS

Steering angle prediction using and 2D image is robust neural network that grasps features to be learnt from images produced by driver to result precise steering gradient or angle values.

scrunity of the results/outputs demonstrates that this model grasps various features that could be understood to a normal human, while taking no notice of infrastructures in the images captured through camera that are not of any importance for driving. This potentiality is acquired from data without the requirement of homebrewed rules and without human intervention for driving.

We have observed and revealed that convolutional neural networks are smart enough to grasp the whole process of lanes and roadways backing without human intervention into roadways or detecting markings of lane, and gain total access over road by generating precise steering angles. A very minor quantity of training image data even less than a few hundred (0100) hrs. of captured driving data was ample to teach the vehicle to drive in vivid conditions, on roadways/highways in various climatically diverse conditions, the neural network model is capable to extract senseful features out of a very scanty training dataset like just steering gradient values or angles.

ACKNOWLEDGEMENT

I would like to thanks my Parents who have enlighten my path with their kind guidance throughout the life.

I would even like to thank Dr.V.V.Mane Principal of TPCT'S College of Engineering, Osmanabad for his guidance and consent to complete this research. I remain thankful to Prof S.A.Gaikwad H.O.D. Dept. of Computer Science & Engineering, for their timely suggestion and valuable guidance.

I am delighted to place on record a special thanks to our PG coordinator Dr.S.N.Holambe for the valuable help in my research work. I am especially thankful of Dept of Computer Science & Engineering for their guidance throughout the research.

VI. REFERENCES

- [1] Mariusz Bojarski (2016). End to End Learning for Self-Driving Cars <https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl->

- using-px.pdf
- [2] Mariusz Bojarski (2017). Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car <https://arxiv.org/pdf/1704.07911.pdf>
- [3] Dean A. Pomerleau ALVINN: An Autonomous Land Vehicle In A Neural Network <https://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network.pdf>.
- [4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [5] Jonas Heylen. From Pixels to Actions: Learning to Drive a Car with Deep Neural Networks https://homes.esat.kuleuven.be/~jheylen/FromPixelsToActions/FromPixelsToActionsPaper_Wacv18.pdf
- [6] anwei Wang and Feng Qi. Trajectory planning for a four-wheel-steering vehicle. In Proceedings of the 2001 IEEE International Conference on Robotics & Automation, May21–262001 URL: <http://www.ntu.edu.sg/home/edwwang/confpapers/wdwicar01.pdf>
- [7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, Winter 1989. URL: <http://yann.lecun.org/exdb/publis/pdf/lecun-89e.pdf>.
- [8] Large scale visual recognition challenge (ILSVRC). URL: <http://www.image-net.org/challenges/LSVRC/>
- [9] Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004. Final technical report. URL: <http://net-scale.com/doc/net-scale-dave-report.pdf>.
- [10] Self-Driving-Car-Simulator <https://github.com/udacity/self-driving-car-sim>
- [11] Google-Waymo-Driverless-Cars-Deep-Learning-Neural-Net-Interview <https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview>
- [12] MIT Deep Learning and Artificial Intelligence Lectures URL: <https://deeplearning.mit.edu/>
- [13] Introduction-Neural-Networks-Deep-Learning <https://www.analyticsvidhya.com/blog/2018/10/introduction-neural-networks-deep-learning/>
- [14] http://street.umn.edu/VehControl/javahelp/HTML/Definition_of_Vehicle_Heading_and_Steering_Angle.html
- [15] <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>
- [16] <https://www.slideshare.net/06Olivier/self-driving-car-nvidia-whitepaper>
- [17] <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>
- [18] <https://www.who.int/publications/i/item/9789241565684>
- [19] <https://www.arxiv-vanity.com/papers/1604.07316/>