A Convolutional Neural Network Approach Towards Self-Driving Cars

Akhil Agnihotri
Department of Mechanical Engineering
Birla Institute of Technology and
Science – Pilani
Hyderabad, India
agnihotri.akhil@gmail.com

Prathamesh Saraf
Department of Electrical & Electronics
Engineering
Birla Institute of Technology and
Science – Pilani
Hyderabad, India
pratha1999@gmail.com

Kriti Rajesh Bapnad
Department of Electrical & Electronics
Engineering
Birla Institute of Technology and
Science – Pilani
Hyderabad, India
bapnadk@gmail.com

Abstract—A convolutional neural network (CNN) approach is used to implement a level 2 autonomous vehicle by mapping pixels from the camera input to the steering commands. The network automatically learns the maximum variable features from the camera input, hence requires minimal human intervention. Given realistic frames as input, the driving policy trained on the dataset by NVIDIA and Udacity can adapt to real-world driving in a controlled environment. The CNN is tested on the CARLA open-source driving simulator. Details of a beta-testing platform are also presented, which consists of an ultrasonic sensor for obstacle detection and an RGBD camera for real-time position monitoring at 10Hz. RRT*-Connect algorithm is used for path planning. Arduino Mega and Raspberry Pi are used for motor control and processing respectively to output the steering angle, which is converted to angular velocity for steering.

Keywords—convolutional neural networks, self-driving cars, machine learning

I. Introduction

As of today, the performance of CNNs has revolutionized image and pattern recognition by surpassing human performance on standard datasets [1]. The strongest feature of CNNs is that they learn features automatically from training examples and hence override the human need to only select intuitive features for the model [2][3]. Another benefit of CNNs is that they take advantage of the 2D structure of images and thus have a higher accuracy than a standard flattened neural network. This approach works better than the explicit feature decomposition approaches such as detection of lanes or neighboring cars, since the network decides the most optimal features to extract from the image for itself. The current models use Gaussian process-based approach to create multi-vehicle interaction scenarios which can create human-like trajectories that fit distinctive street obliges whereas acquiring the key interaction designs. Feedforward Neural Networks, Time Delay Neural Networks, Probabilistic Neural Networks are some of the existing models used in Autonomous Vehicles which form the base of CNN's [4].

This technology provides a robot with senses with which it can traverse in unfamiliar surroundings without damaging itself. This work contains the details of the developed CNN, design of the robot and an experiment proposal to test the autonomy of the robot in a controlled real environment. The autonomous vehicle developed in this project is expected to satisfy the following objectives:

1. To track the lane markings on the road, determine the steering angle and move within the lane accordingly.

2. To have the capacity to detect obstacles along with the speed of obstacle in its path based on a predetermined threshold distance.

3. After obstacle and speed detection, changing its course to a relatively open lane on the road by making a decision autonomously.

4. Plan its path and move accordingly, thus, requiring no human intervention during its operation [5].

II. System Architecture

A. Overview

The dataset given on Udacity's GitHub repository is used for the CNN with 80% used for training and 20% for validation [6]. The training data, in addition to the data from the human driver, consists of images of the vehicle car in various shifts from the center of the lane and rotations from the direction of the road. Time-stamped steering angles are extracted from the .bag files of the training data using Robotic Operating System (ROS) and paired with the corresponding image to form a tuple of (image, steering angle) for input to the CNN, which then computes a proposed steering command.

The system is built using the following (see *Fig. 1 and 2*):

1. Circular metal chassis of diameter 25cm.
2. Two Stepper motors (Nema 17) for the wheels.
3. Two Servo motors attached to the stepper motors to give the steering angle to the vehicle.
4. Ultrasonic sensor HC-SR04 for obstacles detection.
5. Web camera for image capturing of lanes on the road and Raspberry-pi 3B for image processing of the images captured and getting the steering angle.
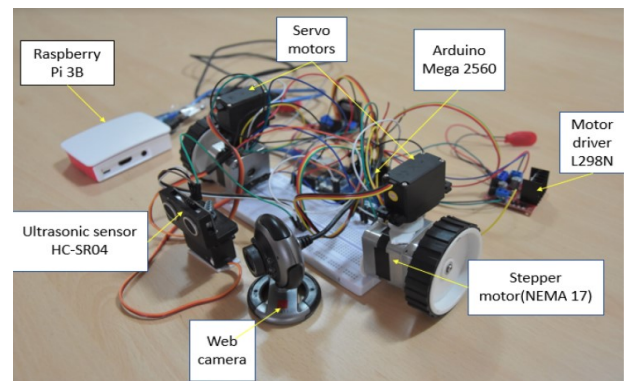

Fig 1: Components of the robot

B. Details of Operation

The network is trained to minimize the mean squared difference of the actual and predicted steering angle as shown in *Fig. 2* [3].

The developed CNN architecture consists of four convolution layers and two fully-connected vanilla layers, each having a different bias, weight and Rectified Linear Unit (ReLU) associated with it. Initially, independent and identically distributed (i.i.d.) weights and bias of 0.1 are
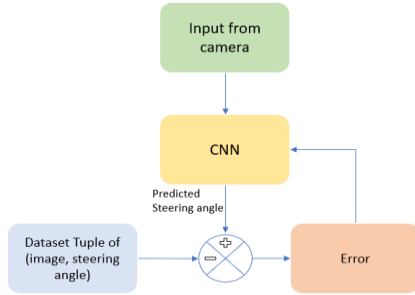

Fig. 2: Overview of the CNN model

assigned. The first three layers are padded with 5x5 kernels with a stride of 2 while the next layer has 3x3 kernels with a stride of 1. These layers are followed by two fully-connected layers which are non-stridden to prevent overfitting on the training data, which is segmented into batches of size 100 and trained for 30 epochs on the CNN. These fully connected layers also serve as a controller for the steering, thus eliminating the use of an explicitly defined controller, normally a Constrained Linear Quadratic Regulation (LQR) controller [7]. *Fig 3*. shows the detailed structure of the CNN.
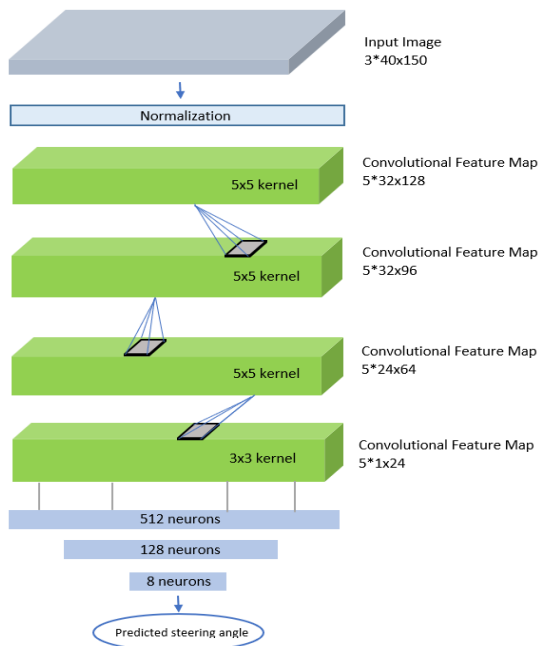

Fig. 3: Architecture of the CNN containing 133,120 weights and about 11 million parameters. In the figure, A*B*C indicates A layers of B width and C length each.

Two servo motors attached to two stepper motors and three ultrasonic sensors are connected to the Arduino Mega 2560 board [8]. The two ultrasonic sensors, for left and right distance measurement are mounted at an angle of 60° from the one at the center of the chassis. Sensor readings and steering angle after image processing by the Raspberry-pi and output of the CNN are fed to the servo motors and the stepper motors by Arduino.

The decision tree behind the code is shown in *Fig. 4* and goes as follows:
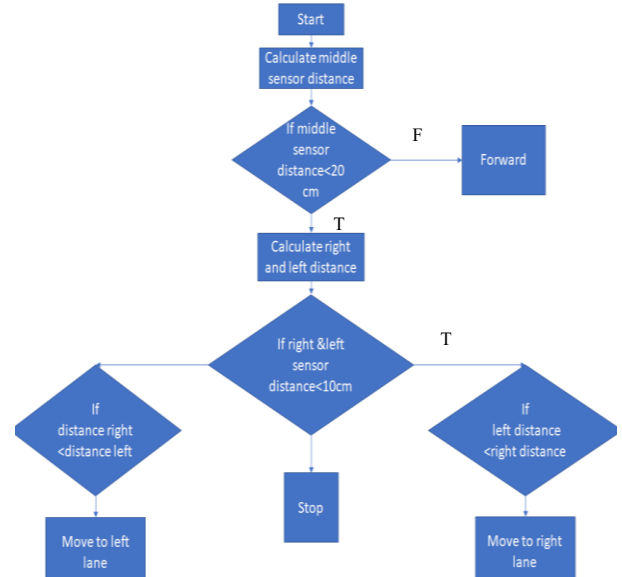

Fig. 4: Decision flow of the sensor input and output. 'T' equals True and 'F' equals False in the figure.

The ultrasonic sensor keeps track of the distance between the car and a possible obstacle in its current lane. For forward movement, the servos on top of the stepper motors point at 90° with the motors running at 60 RPM. If the robot encounters a stationary obstacle (based on the constant rate of decreasing distance), the ultrasonic sensor checks for an obstacle-free left
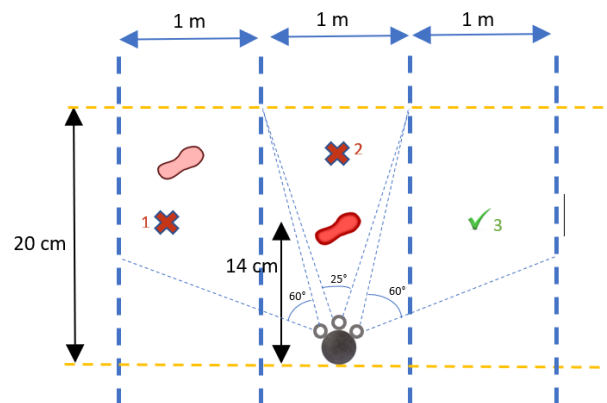

Fig. 5: Schematic showing the robot as a grey disk (bottom center) with three ultrasonic sensors attached to its periphery which divide the line of sight into three zones as shown. In this scenario, there is an obstacle at a distance of 14cm in the current lane, along with another obstacle within 20cm of its current horizontal location. This results in publishing of negative angular velocity of .5 radians per second towards the right after which the CNN takes over to move the robot into lane 3. All values are in SI units.

or right lane for a distance of 20cm and moves to the lane which is obstacle-free at an angle of 60° to the direction of motion. Secondly, if the robot encounters a moving obstacle within 20cm of its current lane, it is programmed to match the speed of that obstacle and continue in its current lane (see *Fig. 5*).

The RRT*-Connect [5] motion planner is used to generate manoeuvres for the vehicle. To generate the trajectories, nearest neighbour calculus based on Reed-Shepp-curves is used, resulting in manoeuvres suitable for confined spaces. The planning algorithm is well-suited for maze-like environments with a lot of obstacles and boundaries.

III. RESULTS AND DISCUSSION

The CNN was trained as described above and tested on a dataset provided by the University of Cambridge to give a sub-par performance since the weather conditions of the two datasets are different [9]. The model was trained on two datasets from Udacity [6] and NVIDIA [10]. The Udacity dataset consisted of 6.6 Gigabytes of 33,478 images while the NVIDIA dataset had 2.1 Gigabytes of 7,064 images. As seen in *Fig. 6*, there is a stark difference in the validation Mean Square Error (MSE) loss of the model after 30 epochs in both the instances visualized with the help of TensorFlow [11]. Udacity dataset gave a loss of 0.0003798 after 30 epochs while NVIDIA dataset gave a loss of 1.013 which was approximately 5 times smaller than the Udacity dataset.

Fig. 7 shows a still from the CARLA simulator in which the CNN was tested and an autonomy (1) of 86.43% was obtained based on the empirical result that human intervention in the simulator was required for five seconds on average.

$$autonomy = \left(1 - \frac{number\ of\ interventions\ .\ 5\ seconds}{total\ elapsed\ time\ (seconds)}\right) . 100 \qquad (1)$$
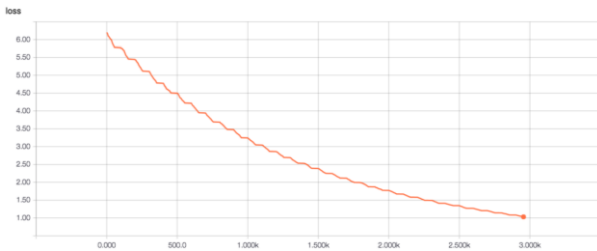
(a)
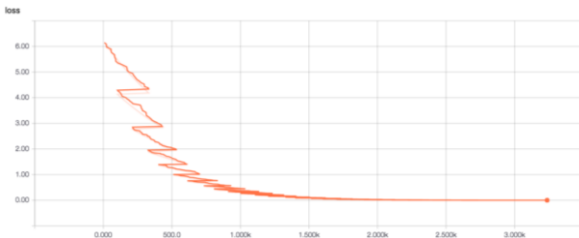
(b)

Fig. 6: MSE over validation data is plotted on the vertical axis and visualized in Tensor Board with number of iterations on the horizontal axis using (a) NVIDIA (b) Udacity dataset. Aside from a lower MSE after an equal number of epochs, the Udacity validation dataset has a larger variance, hence more information, as seen by the erratic fluctuation of MSE in (b).

IV. CONCLUSION AND FUTURE WORK

This paper describes a methodology of implementing a level-2 autonomous vehicle in a relatively sparsely occupied environment. A CNN is trained on a dataset by Udacity and used to compute the optimal steering angle based on the image input through the camera. In case of obstruction in the path, three ultrasonic sensors are used to decide in which direction the vehicle should turn to continue on its path. Once this is achieved, the vehicle resumes its normal functioning of manoeuvring based on the steering angle given by the CNN.

Based on the above results, an experiment is being designed for on-road testing on the intra-campus roads of BITS Pilani Hyderabad Campus. The test-path is 580 meters

Fig. 7: Images from the CARLA Depth camera converted to grayscale and fed into the CNN for lane detection. These images show how the CNN sees the road.

in length with negligible elevation gradient since the dataset was also created on a plane terrain. The path will be marked with three parallel lanes of width 1 meter of Snowcal powder. The GPS module GSM SIM908 will be used for location and path-tracking of the vehicle.

Further research to extend the current CNN to region-based CNNs for posture-recognition passenger safety is also being done. Point RCNN can used to implement 3D object recognition for better path planning and obstacle avoidance. More work is needed to improve the robustness of the network, and to better visualize how the CNN operates internally. Moreover, including other vehicle dynamics parameters in the dataset such as acceleration and velocity will provide for navigation based on real-life constraints on the vehicle.

REFERENCES

[1] 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.

[2] Bojarski, Mariusz & Del Testa, Davide & Dworakowski, Daniel & Firner, Bernhard & Flepp, Beat & Goyal, Prasoon & D. Jackel, Lawrence & Monfort, Mathew & Muller, Urs & Zhang, Jiakai & Zhang, Xin & Zhao, Jake & Zieba, Karol. (2016). End to End Learning for Self-Driving Cars.

[3] 'Trajectory planning for a four-wheel-steering vehicle', Proceedings 2001 ICRA. IEEE International Conference On Robotics And Automation (Cat. No.01CH37164)2001, Danwei WangFeng Qi

[4] 'Types of artificial neural networks', article on wikipedia https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks

[5] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zöllner, R. Dillmann, "RRT*-Connect: Faster Asymptotically Optimal Motion Planning", *Robotics and Biomimetics. ROBIO 2015. IEEE InternationalConference on*, 2015.

[6] Udacity. (2018, February 08). Udacity/self-driving-car. Retrieved from https://github.com/udacity/self-driving-car/tree/master/datasets/CH2

[7] Scokaert, O. M. and James B. Rawlings. "Constrained Linear Quadratic Regulation Pierre." (1998).

[8] Gerard Gibbs, Huamin Jia, Irfan Madani, "Obstacle Detection with Ultrasonic Sensors and Signal Analysis Metrics", Transportation Research Procedia, Volume 28, 2017, Pages 173-182, ISSN 2352-1465, https://doi.org/10.1016/j.trpro.2017.12.183.

[9] Machine Intelligence Laboratory. CamSeq01 ground truth dataset. Retrieved from http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamSeq01/..

[10] NVIDIA. (2018, January 07). NVIDIA Self Driving Car Training Set. Retrieved from https://www.kaggle.com/sameerqayyum/nvidia-self-driving-car-training-set

[11] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[12] Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. "CARLA: An open urban driving simulator." *arXiv preprint arXiv:1711.03938* (2017).