Deep Learning for Self-Driving Cars

Abhijit Vinod Mahalle
117472288
Master of Engineering in Robotics
University of Maryland
College Park, Maryland, USA
abhimah@umd.edu

Pratik Milind Acharya
117513615
Master of Engineering in Robotics
University of Maryland
College Park, Maryland, USA
pratik99@umd.edu

***Abstract*—A convolutional neural network (CNN) has been trained to steer a car in a simulated environment. It maps raw pixels from a single front-facing camera directly to steering commands. The system automatically learns internal representation of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal. Compared to explicit decomposition of the problem, such as lane marking detection, path planning, and control, this end-to-end system optimizes all processing steps simultaneously. This will lead to better performance and smaller systems. Better performance will result because the internal components self-optimize to maximize overall system performance, instead of optimizing human-selected intermediate criteria, e. g., lane detection.**

***Index Terms*—Deep Learning, self-driving cars, CNN**

I. Introduction

CNNs have revolutionized the domain of pattern recognition [2]. Prior to the widespread adoption of CNNs, most pattern recognition tasks were performed using an initial stage of hand-crafted feature extraction followed by a classifier. The advantage of CNNs is that features are learned automatically from training examples. The CNN approach is especially powerful in image recognition tasks because the convolution operation captures the 2D nature of images and learn feature extractor. Also, by using the convolution kernels to scan an entire image, relatively few parameters need to be learned compared to the total number of operations. While CNNs with learned features have been in commercial use for over twenty years [3], their adoption has exploded in the last few years because of two recent developments. First, large, labeled data sets such as the Large Scale Visual Recognition Challenge (ILSVRC) [4] have become available for training and validation. Second, CNN learning algorithms have been implemented on the massively parallel graphics processing units (GPUs) which tremendously accelerate learning and inference. In this project, we have implemented a CNN that goes beyond pattern recognition. It learns the entire processing pipeline needed to steer an automobile. We have used Udacity's self-driving car simulator to collect the data and train the neural network.

II. Related Work

Recent self-driving technology uses two approaches to achieve steering autonomy. They are Mediated Perception approach [5] [6] and End to End learning approach (Behavioral Reflex Approach) [5] [6]. The mediated perception method is based on the human calculated features like road markings [7], path planning [8], obstacle detection [9] and drive control [10]. Different sensors data is required and processed to get these tasks and hence system is not optimal for self driving vehicles. Also, the systems based on Mediated Perception approach are not scalable to different types of scenarios.

On the other side, End to End learning method has been developed to overcome the mediated perception approach. It is also known as behavior reflex approach, which directly map input images to a steering command through End to End approach. This approach is able to provide robust steering commands while removing modeling assumptions and human designed features. ALVINN (Autonomous Land Vehicle In a Neural Network) [11] is the first successful self driving project that uses this approach using Neural Network. This network is simple and it manages to work good on normal roads with less obstacles. After development in the field of deep learning algorithms, Convolutional Neural network plays an important role in self driving technology for steering angle prediction. Nvidia developed a Convolutional Neural Network architecture [1] in 2016 that takes images from the front-facing camera attached to the car as input and gives steering action as output and this project is inspired from the neural network architecture developed by them. They have developed the steering angle prediction as a regression problem. Nvidia found that this approach is very powerful and accurate without using the human control, and managed to drive the vehicle in lane. The CNN model trained by the human driver provided data is insufficient and also CNN model need to learn from miscalculations, hence they have augmented the data by using the three cameras (right, left and center) for training the data for the best steering angle prediction.

III. Methodology

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have

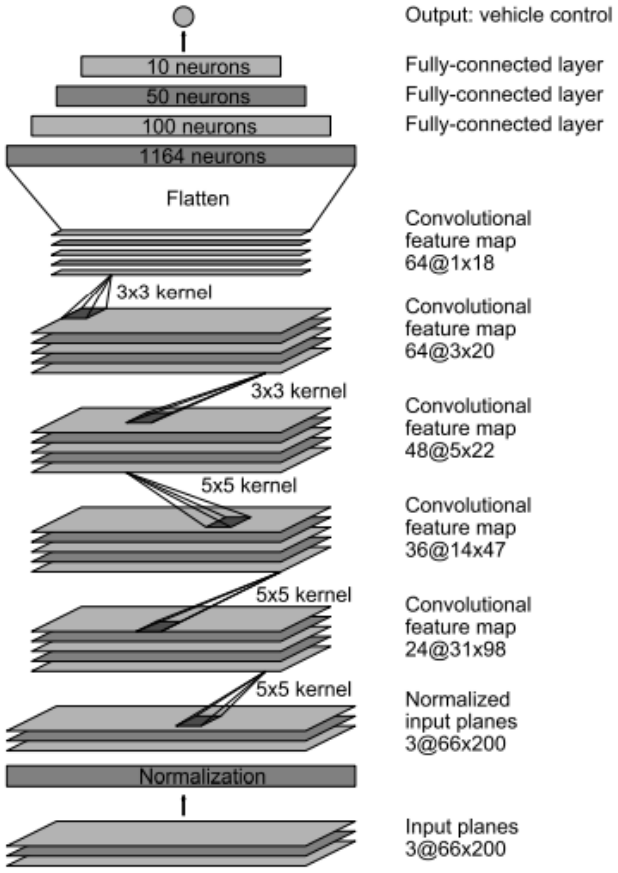

Fig. 1. Nvidia architecture

the ability to learn these filters/characteristics.Convolutional Neural Network are suitable to learn features from images.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

The max pool layer is similar to convolution layer, but instead of doing convolution operation, we are selecting the max values in the receptive fields of the input, saving the indices and then producing a summarized output volume. The implementation of the forward pass is pretty simple.

The Convolutional Network Architecture based on Nvidia's End to End Learning for Self-Driving Cars [1] was used initially (figure 1). The network has about 27 million connections and 250 thousand parameters. However, after training the model based on this architecture, the car was drifting even on straight tracks. Some of the possible reasons may be lack of sufficient data and absence of maxpool layers. Also, this architecture has been designed to deploy in actual cars. Hence, deploying this architecture for simple simulation environment might be an overkill.

A simplified version of the Nvidia architecture (figure 2) was implemented and the results were satisfactory. The network has two convolution layers followed by max pooling and two linearly connected layers. Max pooling is a pooling operation that selects the maximum element from the feature map region covered by the filter. It has total 15 million connections and 120 thousand parameters. The activation function used is Exponential Linear Unit (ELU). The Exponential Linear Unit (ELU) is a neural network activation function. ELUs, unlike ReLUs, it has negative values, allowing them to reduce the computational complexity of batch normalization by pushing mean unit activations closer to zero. The loss function used is Mean Square Entropy Loss and the optimizer used is Adam. The total sample size of the collected data is 24111, out of which 80% i.e. 19288 samples were used for training and the remaining 20% i.e. 4823 samples were used for validating the model. The model was trained using batches of size 32 and the number of epochs run were 22.

The input to the model were images captured by the simulator and the output was the normalized steering angle between -1 and 1. The actual steering angle of the vehicle is between -25° and 25°.

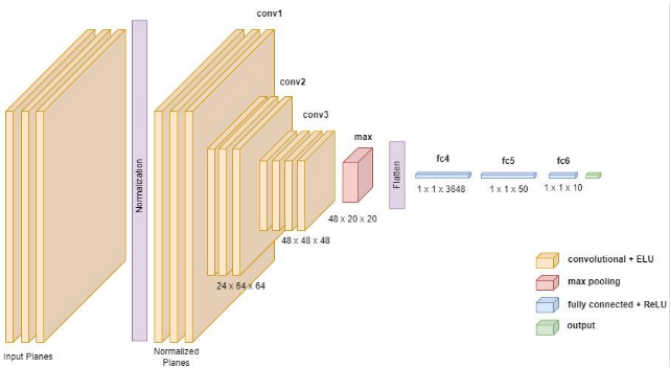

Fig. 2. Simplified version of Nvidia Architecture

IV. Simulation

A. Simulator

Udacity's self-driving car simulator was used to collect the data and test the model. It is an open source simulator based on Unity game engine developed by Udacity in 2016 for its online courses. It has two modes, Training Mode and Autonomous Mode. Training Mode was used to collect the data to train the model and the Autonomous mode was used to test the model. It has two tracks, Beach track and Jungle track. The model was trained and tested on the Beach track.

A simple PI controller was used to control the steering and acceleration of the car. The speed was set to 9 MPH for testing the model.

B. Data collection and Augmentation

The car was driven on the Beach Track for two laps in the Training Mode as seen in figure 2. The car was also driven in the opposite direction to get the reversed turns so that the

Fig. 3. Udacity's self-driving car simulator

Fig. 4. Data collection

model does not overfit the data and can make better left and right turns.

The simulator captures three images for each frame, the center view, the left view, and the right view from the front of the car as seen in figure 3 and stores them in the *data* folder. The simulator also creates a *.csv* file, snippet of which is shown in figure 4. The *.csv* file stores following information for each frame:

1) File location of center view
2) File location of left view
3) File location of right view
4) Steering angle
5) Acceleration
6) Brake
7) Speed

The images stored were a cropped version of the camera, to remove the sky, which was not needed for the training. The stored csv file contained the steering angle for a straight

Fig. 5. Left, center, and right views captured by the simulator

	A	B	C	D	E	F	G
26	C:\Users\Pratik A	C:\Users\Pratik A	C:\Users\Pratik A	0	0.048462	0	5.982447
27	C:\Users\Pratik A	C:\Users\Pratik A	C:\Users\Pratik A	-0.06222	0	0	5.95896
28	C:\Users\Pratik A	C:\Users\Pratik A	C:\Users\Pratik A	-0.38087	0	0	5.897681
29	C:\Users\Pratik A	C:\Users\Pratik A	C:\Users\Pratik A	-0.05994	0	0	5.877697

Fig. 6. csv file snippet

scenario, but the left and right images, which were stored to simulate the car swerving to the right or left, had no steering angle data. For this, the steering angle data from each of the frame was changed by +/- 0.4, for the left and right image respectively, to train the network to take a right turn if it find itself towards the left of the track and so on. The images were also augmented randomly by flipping them horizontally, to make sure that the network does not over-train for only left or right turn. The images were also normalized between -0.5 to 0.5 to improve training.

Fig. 7. Cropped Image

V. Results

As we planned to follow the NVIDIA self-driving network architecture, we replicated the network that was described in the paper. After multiple iterations of testing the trained network with multiple different parameters, the model was unable to follow the road even from a straight path. The car would drift away from the center of the road and would eventually get stuck in the edges of the road. We believed that this was due to the lack of data, as the NVIDIA research paper did mention that they collected 72 hours of real world data for training the network. As we simply lacked the resources to collect that amount of data, and we did not have access to any online database that suited our needs, we were forced to simplify the network to make it work for less amount of training data.

The new network developed did have less convolutional layers, and maxpool layers were added, which provided better result for the same amount of training data. This helped us train the model for one of the two road environments.

Later the model was also fed data from the other environment, which confused the model, and altered the training weights which led to the car being unstable. We believe this was due to the lesser amount of data that we had for the second environment. Hence, we reverted the model back to the old environment data.

Training Loss	
Left Image	0.041
Center Image	0.047
Right Image	0.048
Validation Loss	
Total Loss	0.026

Fig. 8. Losses for the model

The loss result for the training and validation was as such: The video of the final demo can be found here.

VI. Conclusion

The NVIDIA self-driving deep learning architecture was tested in a simulation environment, but due to the results being unsatisfactory, a new network was used, with lesser convolutional layers and added maxpool layers, which showed promising results.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang et al., End to end learning for self-driving cars, 2016.

[2] 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.

[3] L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., , and D Zuckert. Optical character recognition for self-service banking. ATT Technical Journal, 74(1):16–24, 1995.

[4] Large scale visual recognition challenge (ILSVRC).

[5] Sumit Joshi, Narayan Pawar, Vivek Sonara, Sudhir Sul and S. A. Mulay, "End-To-End aDriving Controls Prediction F.Images Using CNN", International Advanced Research Journal in Science Engineering and Technology, vol. 5, no. 3, March 2018.

[6] Z Chen and X Huang, "End-To-end learning for lane keeping of self-driving cars (2017)", IEEE Intelligent Vehicles Symposium Proceedings, pp. 1856-1860.

[7] de Paula, Maurcio Jung and Claudio, "Real-Time Detection and Classification of Road Lane Markings", Brazilian Symposium of omputer Graphic and Image Processing, pp. 83-90, 2013.

[8] J. Yim, M. Kim, S. Shin and J. Park, "Robust path planning for autonomous vehicle in position uncertainty", 2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pp. 608-613, 2014.

[9] N. Deepika and V. V. Sajith Variyar, "Obstacle Classification and Detection for Vision Based Navigation for Autonomous Driving", 2017 International Conference on Advances in Computing Communications and Informatics ICACCI, 2017.

[10] Lie Guo, Ping-Shu Ge, Ming Yue and Yi-Bing Zhao, "Lane Changing Trajectory Planning and Tracking Controller Design for Intelligent Vehicle Running on Curved Road", Mathematical Problems in Engineering, vol. 2014, 2014.

[11] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network in Advances in neural information processing systems", pp. 305313, 1989.