STEERING WHEEL ANGLE PREDICTION FOR SELF-DRIVING CARS

Kiran U Kamath[1], Dhanush B Raj[1], Meghana G S[1], Prajwal T[1], Suhas Bharadwaj R[2]

[1]Dept. of ISE, The National Institute of Engineering, Mysore
[2]Asst. Professor, Dept. Of ISE, The National Institute of Engineering, Mysore

---

Abstract - In this paper, a self-driving car prototype using Deep Neural Network is proposed. Self-driving cars are one of the most increasing interests in recent years. Convolutional Neural Networks(CNNs) have been shown to achieve significant performance in various perception and control tasks compared to other techniques in the latest years. The automated system learns to drive on the local roads with no obstructions and also on the highways with the lane markings on it. This is because of the convolutional neural network (CNN) which is trained to map the pixels from the minimum training data provided by humans. It operates perfectly fine in the areas having hazy visuals like parking lots etc.

With human steering angle as the limited training signal, the system improvises and automatically learns to detect the road visibilities. Despite no explicit training for the detection of outline of the road, the system efficiently optimizes all processing steps simultaneously when compared to explicit decomposition of the problem.

It can be ensured that this will lead to better performance with smaller systems. The overall system performance maximizes by the optimization of system performance. The system learns to solve the problem with a minimum number of processing steps which results in smaller networks.

Key Words: Self-driving car, Deep Learning, Convolutional Neural Networks, Behaviour Cloning, Convolutional kernel, Feature map

1. INTRODUCTION

CNNs have revolutionized the computational pattern recognition process. Before widespread adoption of CNNs, most pattern recognition tasks were performed using an initial stage of manual feature extraction followed by a classifier.

A Convolutional Neural Network is a class of artificial neural networks that uses convolutional layers to filter inputs for useful information. The convolution operation involves combining input data (feature map) with a convolution kernel (filter) to form a transformed feature map. The filters in the convolutional layers (conv layers) are modified based on learned parameters(weights plus bias) to extract the most useful information for a specific task. Convolutional networks adjust automatically to find the best feature based on the task. The CNN would filter information about the shape of an object when confronted with a general object recognition task but would extract the edges of the road when faced with an edge recognition task[1]. This is based on the CNN's understanding that different classes of objects have different shapes but that different types of roads are more likely to differ in shape than in color.

Convolutional networks are composed of an input layer, an output layer, and one or more hidden layers. This allows CNN to transform an input volume in three dimensions to an output volume.

1.1 DATA COLLECTION

We have used data collected from a simulator. While driving a car in a simulator is certainly not the same as driving a car in the real world, there are many similarities. Given the current state of game graphics, images captured in a simulated environment are a good approximation of images that could be captured in the real world. Of course, the simulator also affords safety and ease-of-use. Data collection is much simpler and most importantly, the failed model could cause no threat to life. A successful model might be implemented later in a real car with real cameras.

Our project is mainly based on data that is collected based on behavior cloning[3]. The goal of behavioral cloning is to collect data while exhibiting good behavior and then train a model to mimic that behavior with the collected data. The images that were captured while driving will be fed into a neural network to teach the model how to drive properly.

Each image is captured with the accompanying steering angle. The images will be fed into the network, and the network's job will be to match the appropriate steering angle.

2. MODEL ARCHITECTURE

We train our model to minimize the mean squared error between the steering command output by the model and the values in the dataset[1]. The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers.

The first layer of the network performs image normalization. Normalization normalizes the data between 0–1 by dividing training data and testing data by factor(mostly 255 incase of images). Training and validation set should be normalized with the same factor.

Next 5 layers are convolutional layers which help in feature extraction. We use strided convolutions in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers. Hyperparameters like depth, stride control the size of the output volume. When the stride is 1 then we move the filters one pixel at a time[3]. When the stride is 2 then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially. We follow the five convolutional layers with three fully connected layers leading to an output value of steering wheel.

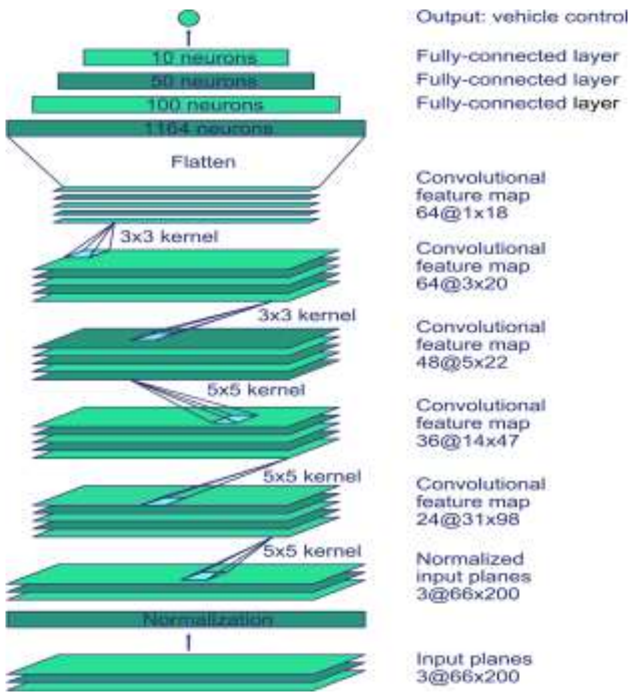

Fig-1 : CNN Architecture

3. TRAINING THE MODEL

3.1 Data Selection

The first step to training a neural network is selecting the frames to use. Our collected data is labeled with road type, weather condition, and the driver's activity. To train a CNN to do lane following we only select data where the driver was staying in a lane and discard the rest. We then sample that video at 10 FPS.

3.2 Augmentation

After selecting the final set of frames we augment the data by adding artificial shifts and rotations to teach the network how to recover from a poor position or orientation. The magnitude of these perturbations is chosen randomly from a normal distribution[5]. The distribution has zero mean, and the standard deviation is twice the standard deviation that we measured with human drivers.

We apply random shear operation to the images generated by the simulator in the preprocessing step. This is done basically to help the car to navigate in the training track.

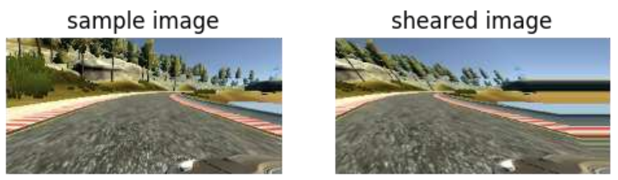

Fig-2: Sheared image after random shaer operation

The images captured by the simulator come with a lot of unnecessary information like the sky, which is not useful for the training of our model. So we crop the useless data from the sheared images.

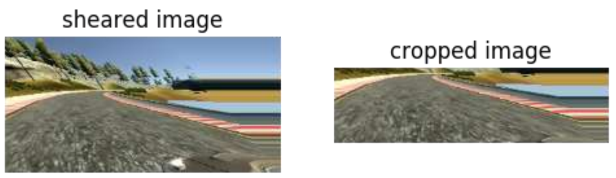

Fig-3 : Cropped image after cropping the sheared image

Next we apply a random flip operation. The idea behind this operation is left turning bends are more prevalent than right bends in the training track. Hence, in order to increase the generalization of our mode, we flip images and respective steering angles.

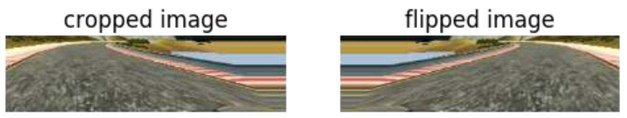

Fig-4 : Flipped image after flipping the cropped image.

4. SIMULATION

Before road-testing a trained CNN, we first evaluate the performance of the network in simulation.

The simulator takes the pre-recorded from the virtual front facing cameras of the car, on the manual driving mode and generates the images. The simulator accesses the recorded test video along with the accurate steering commands during the capture of the video. The simulator sends the first frame of the test video for any shift and rotation and then sends to the trained CNN[4]. Then the CNN returns the steering angle for that frame. Returned steering angles are fed into the dynamic vehicle so that the orientation and position of the simulated vehicle gets updated. The simulator then modifies the next frame in the video so that it appears as if the vehicle were at the position that the steering angle returned by the CNN.

5. EVALUATION

We have the networks provide steering commands in our simulator.

We estimate what percent of the time the network could drive the car(autonamy). The metric is determined by counting simulated human interventions. These interventions occur when the simulated vehicle departed from the centreline by more than 1m[3]. We assume that in real life an actual intervention would require a total of 6second: This is the time required for a human to retake control of the vehicle, recenter it, and then restart the self steering mode. We calculate the percentage autonomy by counting the number of interventions multiplying by 6 seconds, dividing by the elapsed time of the simulated test, and then subtracting the result from 1:

$$Autonomy = \left(1 - \frac{(no.\,of interventions).\,6sec}{elapsedtime[seconds]}\right).\,100$$

Suppose, if we had 20 interventions in 1200 seconds, we would have an autonomy value of

$$\left(1 - \frac{20.6}{1200}\right).\,100 = 90\%$$

6. CONCLUSION

We have successfully shown that CNNs are able to understand the entire learning process lane and road following without manual decomposition into road or lane marking detection, path planning, and control. A small amount of training data from less hours of driving was sufficient to train the virtual car to operate in diverse conditions, on highways, local and residential roads in sunny, cloudy, and rainy conditions[4]. The CNN is able to extract the meaningful and useful road features from a very sparse training signal(only steering). For example our autonomous system learns to detect the outline of a road without the need of explicit labels during training.

As can be seen, there are many areas we could explore to push this project further and obtain even more convincing results. Furthermore, we are going to take throttle into the model with the ambition of achieving higher levels of autonomous cars.

REFERENCES

[1] Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

[2] S. Ren, K. He, R. Girishick, and J. Sun, "Faster R-CNN:towards real time object detection with region proposal networks," Advances in Neural Information Processing Systems(NIPS), 2015.

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Fleep, P. Goyal, L. D. Jackel, M.Monfort, U. Muller, and J. Zhang, "End to end learning for self-driving cars," 2016.

[4] Advanced Applied Deep Learning ,Convolutional Neural Networks and Object Detection, Umberto Michelucci, Dec 26,2019

[5] Hands-On Machine Learning with Scikit-Learn and TensorFlow by Aurélien Géron