Prediction of Steering and Throttle Commands for Self - Driving

Shivani Reddy Rapole srapole@seas.upenn.edu
Shreyas Ramesh shreyasr@seas.upenn.edu
Vasanth Kolli vasanthk@seas.upenn.edu

ABSTRACT

In this project, we try to predict steering commands: steering angle and throttle by directly using the images (of the road and surroundings) from cameras mounted on the vehicle as inputs. We aim to achieve this by applying an end to end learning approach. We trained multiple models on the real world data and simulation data from Udacity Car simulator to obtain the desired outputs. We evaluate the performance of our final network architecture by using the model outputs to autonomously steer a vehicle in a simulated environment.

INTRODUCTION

Autonomous driving has been a fascinating challenge for engineers and computer scientists alike. A lot of research has gone into the world of autonomous driving, its innumerable problems, the immense potential of the field when it comes to safety, efficiency and economic feasibility. Billions of dollars are spent on research and development of autonomous mobility solutions and to achieve the gold standard of self-driving, level 5 autonomy.

The early stages of autonomous driving involved very expensive sensors such as LiDAR, Radar and ultrasonic sensors. But companies such as Tesla, are trying to double-down on vision rather than sensor fusion for autonomous driving, eventually running on a set of finite number of video cameras. Such an approach requires a lot of data as well as many powerful systems to train the model using these data to ensure that the system is safe for public use.

One of the biggest challenges to autonomous driving research is the real-time responses. The environment is continuously changing and the model needs to keep up with the changing states in order to manoeuvre the vehicle safely.

RELATED WORK

U. M. Gidado et, al [2] provided an extensive survey on the different frameworks, simulators, as well as a comparison study between CNN and Deep Reinforcement Learning models. It is observed that convolutional neural networks are very important in the domain and more heavily relied on by researchers than any other DL architecture. It also outlines the various challenges faced by researchers when trying to solve the problem of steering prediction using the various models, such as lack of sufficient data, high computational and memory costs as well as poor performance at unknown surroundings.

The project is inspired by M. Bojarski *et al* [1] which was published by Nvidia in recent years. The goal of Nvidia was to build an end-to-end learning system which can directly learn steering commands from pixel input based on a single central image. The model is trained using 72 hours of real-life driving in many cities and highways.

Udacity's Car simulator repository [3] as well as Naoki Shibuya's article [4] provided the groundwork for simulation as well as the method of running our model in the simulation and evaluating its performance.

METHODOLOGY

Dataset

1. Simulation

We are using Udacity's self-driving car simulation for training and simulation. The training mode allows us to manually drive the vehicle along the track using the keyboard, and at every second three images, (left, centre and right) are generated with a resolution of 320x160. The simulator also generates a CSV file; it consists of the location (file system path) of the three images, steering angle, throttle and vehicle speed.

The autonomous mode in the simulator helps us run our model to evaluate the performance of our trained model. It generates the center image that is then sent to the model, the model computes the steering angle and throttle which is sent back to the simulation which finally steers and moves the vehicle accordingly.

2. Real-life

We use the dataset provided by Sully Chen's github repository [7], that has close to 63,000 images. And stores the image as well as the steering angle of the vehicle at that instance. This dataset is used to train the model for steering angle prediction.

Model Architecture

We implemented the CNN architecture from scratch, which takes images from camera(s) mounted on a vehicle as input and predicts the steering commands like steering angle and throttle as a supervised learning task.

As shown in Fig.1, the CNN architecture we implemented has a stack of five convolutional layers followed by four linear layers. The activation function used for all the layers is ReLU. We used a learning rate of 0.001, MSE loss function and Adam optimizer to train our model. We trained this network to learn steering angle given a single image input. We used the same architecture to train for throttle prediction given the single central image input.

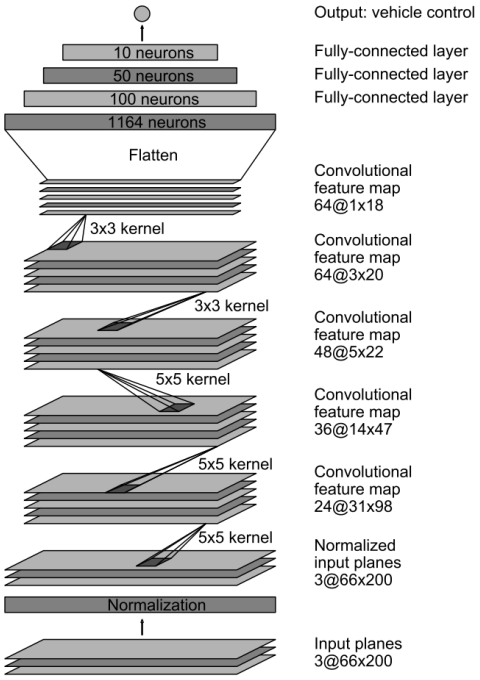

Figure 1. CNN Architecture from NVIDIA Paper

Further, we researched for different possible architectures which could predict both steering angle and throttle in one go. We came across the concept of multi-task learning which is based on shared layers between two correlated tasks. Since, both steering angle and throttle require a high level understanding of the image frame like detecting edges in the beginning before learning any task-specific convolutional kernels, we believed that this would be the approach best suited to our purpose. So, we designed our own model based on the layers presented in our previous approach.

This is also mainly inspired by the current trend in autonomous driving where multi task learning is one of the most preferred architectures. This approach in autonomous driving incorporates a variety of tasks from moving object detection, traffic sign detection, lane marking detection to name a few in their multi task models.

As shown in fig.2, we structure the first three 2D Convolutional Layers similar to our previous model, which learn the low level features, as shared layers in our multi-task model. We structured the remaining two Convolutional layers and all the linear layers for throttle and steering angle prediction as task specific layers. These individual layers will learn the task-specific parameters while training. Backpropagation with respect to both the tasks simultaneously will help the shared layers to learn better convolution filters, which would be the main advantage of multi task learning.

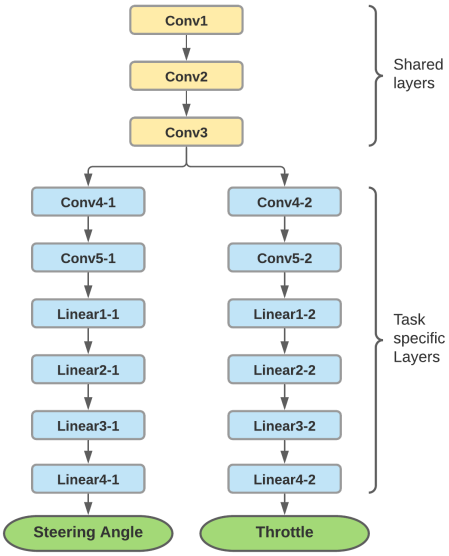

Figure 2. Multi-Task Model Architecture

Apart from tuning the model architecture to suit the purpose, we experimented with different approaches for the inputs and analyzed the performance of our model:

(a) The basic way is to use the central image as input

(b) The second way is to stitch the left, central and right images together and send the stitches image as input to the model

RESULTS

First, we trained the CNN architecture for steering angle prediction using the real driving image data and simulated data. We also tried the Multitask learning approach, and interpreted the activation outputs from the Conv layers of our model to understand the features which the model deems to be salient features for the steering controls prediction task.

CNN for Steering Angle Prediction

We resize the images to focus on the main content i.e. road, lane markings etc. We tuned the hyperparameters like learning rate, number of linear layers etc to stabilise the training process and achieved a stable decrease in the training and validation losses as shown in fig.3.

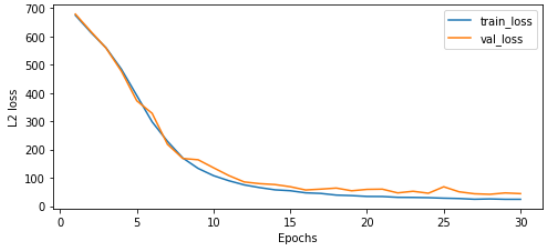

Figure 3. MSE Loss curve of CNN architecture for steering angle prediction using real driving data

We used the predictions of this model to autonomously steer and drive the car in the Udacity Driving Simulator. As the model was trained on real images, and the testing was done on simulated images, we observed that there is a context mismatch and so it did not perform too well on the simulation. To improve our results, we recorded the runs on a simulator and made a new dataset containing close to 30k images. This dataset has left, centre and right images along with steering angle and throttle values.

We resized these images and trained two CNN models which are intended to predict steering and throttle independently. We observed that, in comparison to the original real world dataset, the

simulated dataset provided features which were relatively of lesser value for the CNN. Fig.4 shows the loss curves obtained during the training of CNN architecture to predict the steering angle based on Udacity simulated dataset.

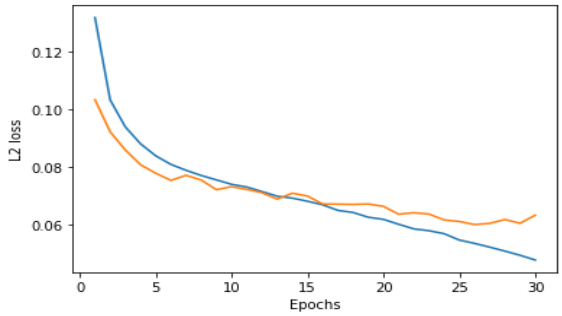

Figure 4. MSE Loss curve of CNN model for steering angle prediction using simulated driving data; Train loss (in blue) and Validation loss (in orange)

To further improve the performance during autonomous driving, we tried multiple modifications on different components of the network architecture, dataset augmentation and other datasets. Below listed are some of the modifications:

(a) Using ELU activations after each layer instead of ReLU

(b) Incorporating Batch Normalisation

(c) Using dropouts

(d) Tried different architectures by changing the parameter (kernel size, stride) of convolutional layers and changing the number and size of Fully Connected layers.

(e) Augmented images by random horizontal flipping of input images changing steering command labels appropriately.

(f) Tried running our model on Kaggle dataset which was generated using the Udacity simulator.

All the approaches mentioned above were giving similar results and we did not observe any significant shift in performance by employing above methods on the task of steering angle prediction.

Multitask Learning

Later, we studied the performance of the multi-task learning approach to predict steering angle and throttle from images. It should be noted from fig.5 that the multi-task model converged much faster (10 epochs) than the normal CNN architecture (30 epochs) in fig.4. We effectively reduced the training time by switching to this approach. This was expected behavior since the multi-task model is a stronger model, and it has two directions for back propagations to go to the initial layers, through steering angle and throttle columns for every data input.

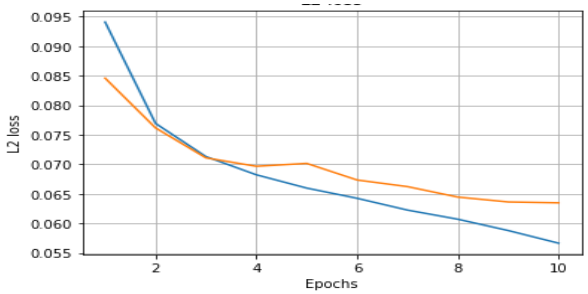

Figure 5. Loss curves of Total MSE loss (Average of Steering angle loss and Throttle MSE loss) , Train loss (in blue) and Validation loss (in orange)

Image Stitching

We explored the idea of stitching the images from the left, center and the right cameras and subsequently use the stitched images to train the model. The idea here was that the stitched images would contain more information regarding the environment surrounding the vehicle and thus provide better steering angle and throttle prediction results.

The stitching operation was done in two steps, first we stitch the left camera image and the center camera image. Followed by stitching the stitched output from the first step with the right camera image. Surprisingly, for a lot of the (left, center and right) image triplets, there were not enough

correspondences to stitch the images. This might be due to the fact that in the simulation environment there is a lot of similarity between the left, center and right images and thus identifying the best match in the target image for a given feature in the source image was difficult. Also, in the images that were stitched we observed that the images were skewed towards the left camera image as the stitching is done in a panoramic style starting from the left camera image. Due to a high variance and skew in the stitched images output and failure to stitch a lot of the images due to correspondence failure issues mentioned above, the stitched image dataset was deemed not suitable for training a CNN network.

Figure 6. Left skew in stitched image

Interpretability of Convolutional Layers

To interpret which kind of features the kernels of our convolution layers are internally learning, we analysed the activation outputs of different channels across various layers of our model. We decided that only the outputs of the first two layers among our five convolutional layers are visually interpretable because the size of each channel in terms of height and width will reduce as we delve deeper into the network.. In fig.7, we show the original image, activation outputs of a few channels after the first convolutional layer, and the result of canny edge detection on the input image for comparison (as edge detection and segmentation are some of most common tasks required to analyse any image). It can be observed that the output channel of the first layer does something similar to segmenting the main object (road) and background (grass and trees) while simultaneously highlighting important features like the road and lane markings. We also observed that the eight channel and nineteenth channel detect horizontal and vertical edges respectively. A single convolution layer is capable of learning to capture details to a huge extent, the remaining layers attend to more task-specific features. This clearly demonstrates the power of Convolutional Neural Networks in the domain of Computer Vision.

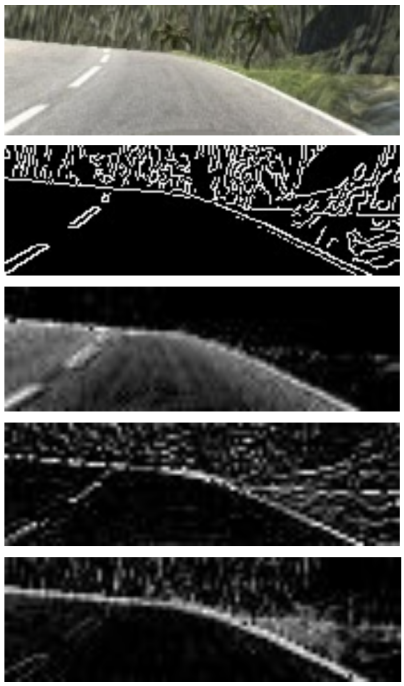

Figure 7. Images in order (a) original RGB image input (b) Canny Edge detection output (c) ReLU Activation output of first channel in Conv1 output (d) ReLU Activation output of eighth channel in Conv1 output (e) ReLU Activation output of nineteenth channel in Conv1 output

CONCLUSION

Our end goal of this project was to autonomously steer the vehicle based on the outputs (steering commands) given by our model in the Udacity simulation environment.

The model with only steering angle as output performed reasonably well in the Udacity simulation. The model had slight bias towards

steering left as the simulation tracks from which the data was generated had a majority of left turns. The multitask learning model converges faster in training when compared to the model with only steering angle as input.

When we analyzed the autonomous driving performance, we found out that there is an issue with the Udacity dataset - the throttle value is never negative. So, after driving for a while the car gradually deviates from its path due to compounded positive throttle values. But, this did not occur when we predicted the steering angle based on a real driving dataset and calculated the throttle based on a physics equation. So, our model definitely had the capacity to learn the steering commands if better data had been available.

FUTURE WORK

While this focused on steering and throttle prediction in simulation, some of the tasks mentioned below can be implemented to augment our current study:

1. Pretrain multi-task on real driving data and tune it for the simulated dataset.
2. Send 3-images as input to the network that are left, center and right camera images as inputs [input dimension size (200x66x9)].
3. Add object detection task to multitask learning network.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude towards Prof. Jianbo Shi, our team mentor Anirudh Subramanyam and the entire teaching staff of CIS 581 for their guidance throughout the course of this project.

REFERENCES

[1] M. Bojarski *et al.*, "End to end learning for self-driving cars," *arXiv.org*, Apr. 25, 2016. https://arxiv.org/abs/1604.07316

[2] U. M. Gidado, H. Chiroma, N. Aljojo, S. Abubakar, S. I. Popoola, and M. A. Al-Garadi, "A Survey on Deep Learning for Steering Angle Prediction in Autonomous Vehicles," *IEEE Access*, vol. 8, pp. 163797–163817, 2020, doi: 10.1109/access.2020.3017883.

[3] Udacity, "GitHub - udacity/ self-driving-car-sim: A self-driving car simulator built with Unity," *GitHub*. https://github.com/udacity/self-driving-car-sim (accessed Dec. 17, 2021).

[4] Naoki, "Introduction to Udacity Self-Driving Car Simulator - Naoki," *Medium*, Mar. 05, 2017. Accessed: Dec. 17, 2021. [Online]. Available: https://naokishibuya.medium.com/introduction-to-udacity-self-driving-car-simulator-4d78198d301d

[5] brianpinto91, "GitHub - brianpinto91 /image-stitching: A python package to stitch multiple images either horizontally or vertically," *GitHub*. https://github.com/brianpinto91/image-stitching (accessed Dec. 17, 2021).

[6] D. Soni, "Multi-task learning in Machine Learning," *Towards Data Science*, Jun. 28, 2021. Accessed: Dec. 17, 2021. [Online]. Available: https://towardsdatascience.com/multi-task-learning-in-machine-learning-20a37c796c9c

[7] SullyChen, "GitHub - SullyChen/driving-datasets: A collection of labeled car driving datasets," *GitHub*. https://github.com/SullyChen/driving-datasets (accessed Dec. 17, 2021).

[8]Z. Na, "Self-Driving Car Simulator," *Kaggle*. https://www.kaggle.com/zaynena/selfdriving-car-simulator (accessed Dec. 17, 2021).