# Improving steering angle prediction accuracy in autonomous vehicles using lane detection

Abhishek Singhal
abhsinghal@umass.edu

Dilip C Kavarthapu
dkavarthapu@umass.edu

## Abstract

*Self driving cars are on the rise today and expected to become mainstream few years down the line. One of the key subsystems of such vehicles to become successful is that of a robust steering angle prediction to navigate the vehicle in an organized manner. Udacity has released a dataset of images along with steering angle while driving. We show an improvement on this prediction compared to several other solutions by using augmented data generated from a successful lane detector deep neural network.*

## 1. Introduction

Self driving cars are expected to be one of the next big revolutions in the automobile industry. It is expected to generate enormous amounts of value and optimization of natural resources over a period of time. Multiple companies have invested aggressively in this area in order to develop the best such systems.

In the pursuit of solving the challenge, one of the key sub problem to be solved is that of creating a robust steering angle prediction model. Udacity, with the intention of building the world's first open source self driving car platform has released a set of online challenges that aim to solve different parts of the platform. We attempt to provide the best solution to the second challenge that was released which deals with task of predicting steering angles using deep learning.

As a part of the challenge, a dataset was released that contained images along with the correct steering angle captured while driving. There have been tons of submissions attempting to solve this better. However, the motivation for our attempt at this challenge is that the problem can be segmented into 2 parts : lane detection and steering angle prediction and that the latter is dependent on the former. We approach this problem to improve the predictions by augmenting data obtained from detecting lanes from an optimally trained lane detection module and use it on few of the best architectures and methods that have already been attempted. We also want to test the hypothesis if the pre-trained lane detection module is going to help in improving the results on the steering angle prediction.

The remainder of the paper is organized as follows: section 2 describes related works, section 3 describes the approach, section 4 presents the experimentation, section 5 describes the results and section 6 describes the conclusion.

## 2. Related Work

In 2016, Nvidia demonstrated [2] that CNNs can be applied in prediction of steering angles. They steered an autonomous vehicle using CNNs in real time setting on highway lanes. Three cameras were mounted in front of car, however they have used the center camera only to steer the vehicle well.

Udacity dataset also contain similar structure (left, center, right cameras are mounted) and driving information such as steering angle, speed, torque etc. are recorded. This open source challenge generated a lot of contribution and innovations while solving this problem. Apart from this challenge, there are few other challenges released by Udacity in this domain as well.

Some submission papers [8], [4], [5] tried to incorporate temporal information (time-series), building on top on Nvidia model which processes single images. The ideas were based on on the fact that the driving process is continuous and that the predictions must be smooth. The idea of having temporal information can

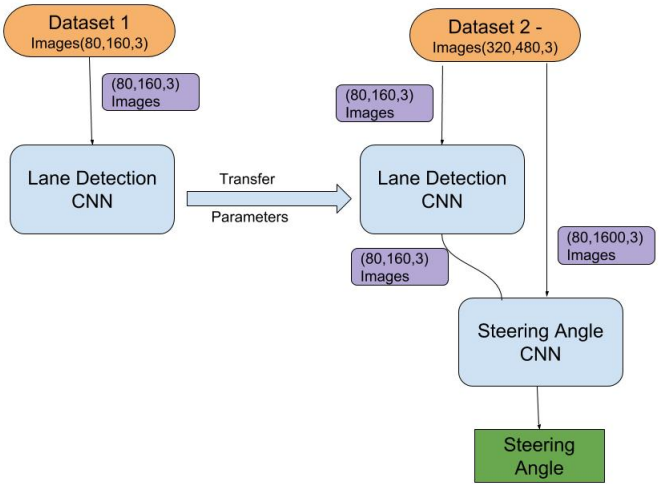


Figure 1. Representational graph of the flow of our approach

Figure 2. A good image from Udacity dataset

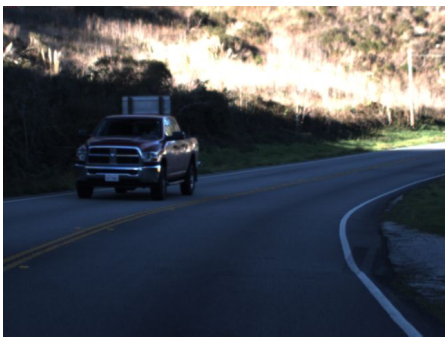

Figure 3. Another good image from Udacity dataset

help in that case because you have the prediction on the previous frame which can help the model predict a value close to it.

Furthermore et al.[3] develop a novel recurrent convolutional architecture suitable for large-scale visual learning. This model was end-to-end trainable, and demonstrated good performance on benchmark video recognition tasks. The ideas mentioned here in the encoder part where the convolutional networks were used can be helpful in developing better recognition models in this domain.

Wu et al.[7] trained videos using multi-stream deep networks involving both CNN and LSTM. Simonyan and Zisserman[6] proposed Two-Stream Convolutional Networks for action recognition in videos, fusing outputs of a spatial and motion stream CNN. This is more similar to the ideas we want to implement in this project.

## 3. Approach

We divide the problem into two parts : lane detection and steering angle prediction and try to verify if the former can help improving the results of the latter. The idea is that we first train the lane detection module on the appropriate dataset and fix it's parameters. Later, the steering angle prediction network is trained on an appropriate dataset where we also augment data to it's inputs by passing it through the previously trained lane detection module. This approach is shown in Figure 1.

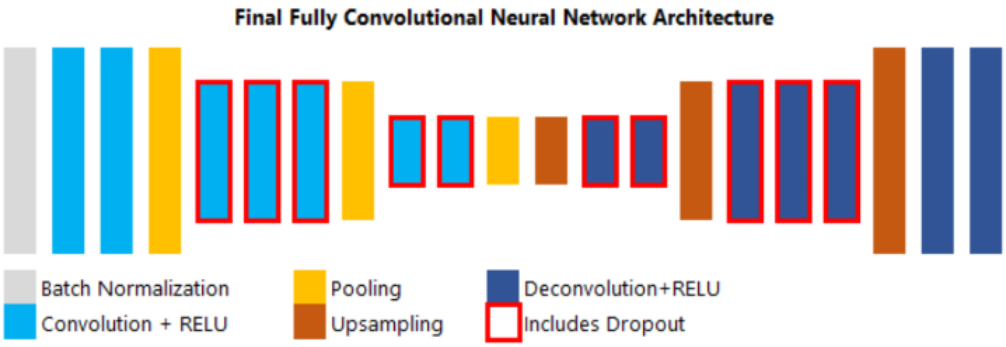


Figure 4. Lane detector CNN architecture

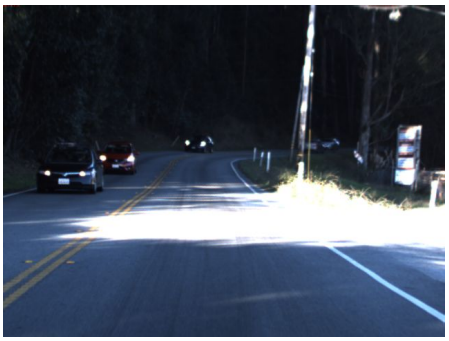

Figure 5. A dark bad image from Udacity dataset

Figure 6. Another dark bad image from Udacity dataset

So, trying to define it mathematically, let F be the model that is trained on the lane detection dataset, F(i) be defined as the output of the model F on image i and $M_x$ be defined as the model of the steering angle prediction that is trained on the dataset x. We thus want to test the hypothesis that will $M_{x,F(x)}$ perform better that $M_x$ or not.

### 3.1. Datasets

For this project, we use two datasets - one for the lane detection module and the other for the steering angle prediction module.

**Dataset 1** : The dataset is used for the lane detection module and consists a total of 12,764 images of size (80,160,3). The output labels are images with lanes marked in green channel.

Some statistics about the data :-

- 26.5% were straight or mostly straight roads, 30.2% were a mix or moderate curves, and 43.3% were very curvy roads
- 17.4% were clear night driving, 16.4% were rainy morning driving, and 66.2% were cloudy afternoon driving

**Dataset 2** : The dataset is used for the steering angle prediction module and was taken from Udacity's self-driving car challenge 2. While the input labels are images of the camera in the front, the output is a scalar which is the correct steering angle. Sample good images are shown in Figure 2 and Figure 3 and some dark images are shown in Figure 5 and Figure 6. The following are some statistics about the dataset :-

- Clip 1: 221 seconds, direct sunlight, many lighting changes. Good turns in beginning, discontinuous shoulder lines, ends in lane merge, divided highway
- 791 seconds, two lane road, shadows are prevalent, traffic signal (green), very tight turns where

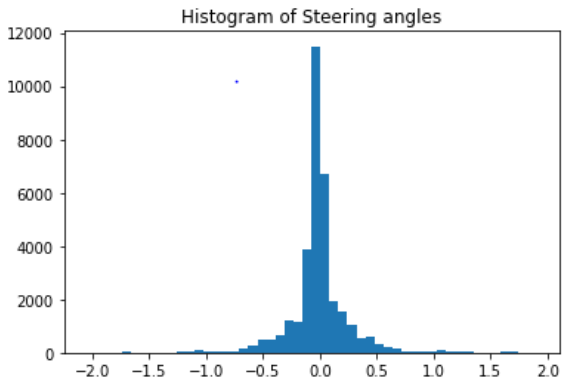


Figure 7. Histogram of Steering angles

center camera can't see much of the road, direct sunlight, fast elevation changes leading to steep gains/losses over summit. Turns into divided highway around 350s, quickly returns to 2 lanes

- Clip 4: 99 seconds, divided highway segment of return trip over the summit
- Clip 5: 212 seconds, guardrail and two lane road, shadows in beginning may make training difficult, mostly normalizes towards the end
- Clip 6: 371 seconds, divided multi-lane highway with a fair amount of traffic

We plotted histogram of steering angles in the dataset, and found that a large number of labels were neutral angles (+- 0.050) . This affects the CNN prediction since the model is biased towards neutral angles. To overcome this, we removed frames within (+-0.010) with 40% probability.

**Preprocessing** : As a part of preprocessing on the datasets we gathered, we have tried the concept of data augmentation by flipping on dataset 1. This means that for every image in the dataset, we have added a horizontally flipped input and output of the same image as the nature of the problem allows us to do that. This allows us to have additional valid dataset and can help in getting better performance.

In order to deal with concentrated data with outputs near zero in dataset 2 and create a more uniform dataset, we removed a number of samples with output in the range [-0.05, 0.05] with probability 0.4.

### 3.2. Lane detection module

The objective of this module is to detect the lane on the road on which the vehicle is travelling given an image facing towards the road taken while driving. we have used a deep learning architecture inspired from : [1] about which we describe below.

The architecture uses a combination of convolutional layers with RELu, batch normalization, pooling, upsampling and dropout layer, and is shown in Figure 1. The network used a filter size of 3x3 and max pooling layer of size 2x2 with a total parameter count of 181, 693.

### 3.3. Steering angle prediction module

The objective of this module is to be able to predict the correct steering angle given an image facing towards the road taken while driving . Apart from just using this image directly, we detect the lane using the model mentioned above and append this output image to to the original one before passing it through the deep network to predict the steering angle.

We use the following 2 architectures for the steering angle prediction module and combine it with the lane detector module:-

- Baseline architecture : This is very similar to lane detector architecture, except that deconvolution layers are replace by 3 fully-connected layers which would then predict a single value.
- Nvidia's architecture : This architecture can be seen in Figure 7. The filter size is 5x5 for first 3 layers and 3x3 for other layers while the max pooling layer is of size 2x2. Fully connected layers are regularized accordingly. The total number of parameters in this network are 11, 502, 851.

## 4. Experimentation

For the lane detection module, we optimized the MSE loss over the pixel values on the predicted image by using the Adam optimizer with learning rate 1e-3. As for the steering angle prediction module too, we used the Adam optimizer with a learning rate of 1e-3 by optimizing on the MSE loss of the predicted steering angle scalar.

MSE loss in the case of lane detection for a model F can be written as

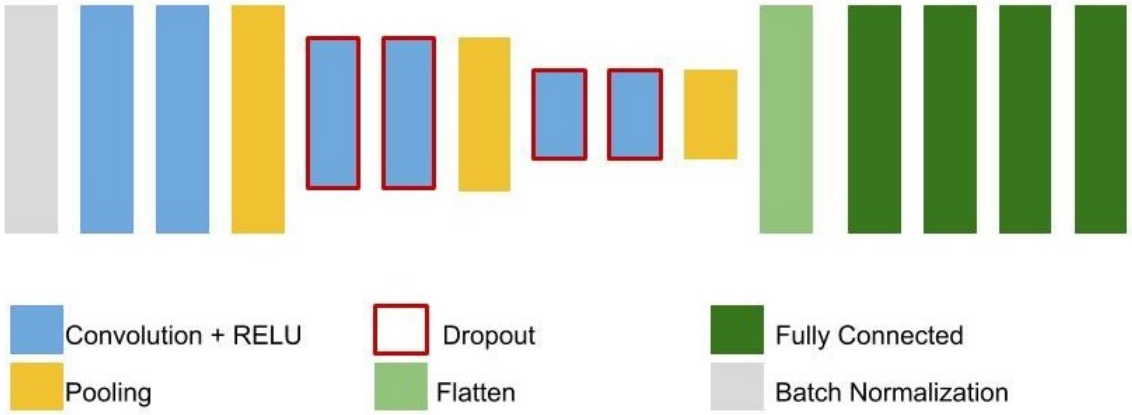


Figure 8. Steering control architecture by Nvidia

$$MSE_{F,x,y} = \sum_i \sum_j (y - F(x)_{i,y})^2$$

where F is the model which predicts the output lane detected image on on image x and y is the actual lane image.

Similarly, in the case of the steering angle prediction for a model M, the MSE loss can be written as

$$MSE_{M,x,y} = (y - M(x))^2$$

where M(x) is the steering angle prediction made by the model M on image x and y is the ground truth value.

We thus perform a total of 4 experiments on the steering angle prediction problem - baseline architecture with the lane detection module, baseline architecture without the lane detection module, Nvidia's architecture with the lane detection module and Nvidia's architecture without the lane detection module and report the results accordingly in Table 1. Even though the loss that has been optimized is the MSE loss, we report the RMSE in order to compare with the results on the Udacity leaderboard.

RMSE in this case can be written as follows

$$RMSE_{M,x,y} = \sqrt{\frac{\sum_{i \epsilon N} (y_i - M(x_i))^2}{N}}$$

where N is the set of all test images and other parameters are same as defined above.

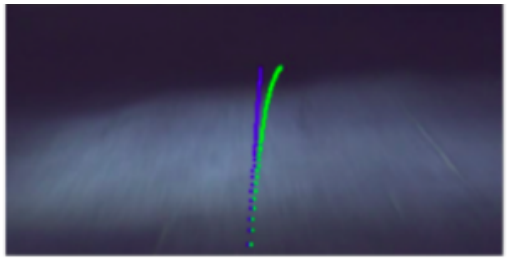

Figure 9. Night driving prediction

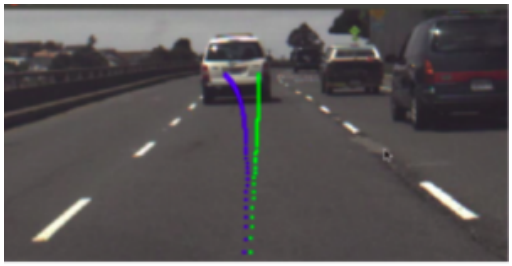

Figure 10. Prediction on highways

## 5. Results

As can be seen in the table, in the case of the baseline architecture, the performance of the model improved clearly from using a lane detection module when compared with not using the lane detection module. On the other hand, for the Nvidia's architecture, we don't observe a significant change in performance

Figure 11. Prediction at intersections

| Method | Train Set | Val Set | Test Set |
|---|---|---|---|
| A1 w/o ld | 0.0371 | 0.0512 | 0.0972 |
| A1 with ld | 0.0359 | 0.0464 | 0.0756 |
| A2 w/o ld | 0.0690 | 0.0757 | 0.1057 |
| A2 with ld | 0.0600 | 0.0656 | 0.1026 |
| Best model | | | 0.0512 |

Table 1. RMSE values on test dataset with different models (ld stands for lane detection. Best model refers to the top model on the Udacity leaderboard.

on the test data. However, we can observe that for the validation data, the performance has improved clearly. From these results, we can infer that the hypothesis we were testing is true.

If compared to the results on the leaderboard of the Udacity self driving car challenge 2, all the models that we trained would have been in the top 10. However, the best performing model of ours we observed was the baseline architecture with the lane detection module.

Another significant feature in this project is that fact that even though there is a domain shift from the 2 problems that were aiming to optimize, the results of a model trained on 1 dataset and tested on the other are quite good. For example, in Figure 12, we can see the picture of the quite accurate lane prediction of the model has been trained on dataset 1 and tested on an image from dataset 2.

## 6. Conclusion

We have empirically tested and see that our hypothesis is true. This shows that rather than simply applying a complex neural network to some data to get the results, it might be helpful to be able to break down the problem into multiple parts, solve them perfectly and append them together to bring out the best results as

Figure 12. Test image from Udacity dataset

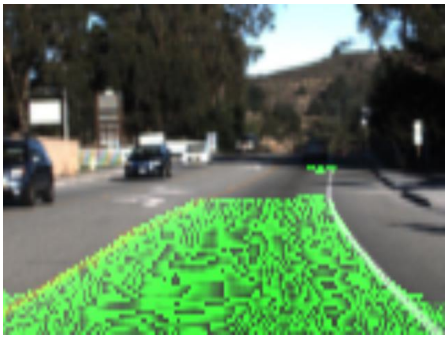

Figure 13. Lane detection on test image

can be seen in the case of the experiments performed above.

On the other hand, despite having shown an improvement in the performance, there still is lot of scope to make the model better. In our project, we have not considered dealing separately with issues like extra lighting and shadows which might affect the performance of the model. We can in future try to use computer vision techniques to deal with such problems.

There is also scope for us to try out different and innovative architecture like the 3d convolution networks where we also consider the temporal frames to predict the steering angle. Given that it is temporal, there is scope for it to behave more continuous with less jitters.

## References


[1] https://github.com/mvirgo/mlnd-capstone.

[2] 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. *arXiv preprint arXiv:1604.07316*, 2016.

[3] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[4] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[5] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[6] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[7] Z. Wu, Y.-G. Jiang, X. Wang, H. Ye, X. Xue, and J. Wang. Fusing multi-stream deep networks for video classification. *arXiv preprint arXiv:1509.06086*, 2015.

[8] Z. Wu, T. Yao, Y. Fu, and Y.-G. Jiang. Deep learning for video classification and captioning. *arXiv preprint arXiv:1609.06782*, 2016.