Improving steering angle prediction accuracy in autonomous vehicles using lane detection

Dilip C Kavarthapu
University of Massachusetts Amherst
dkavarthapu@umass.edu

Abhishek Singhal
University of Massachusetts Amherst
abhsinghal@umass.edu

Abstract

One of the key sub systems in a self driving car is that of the steering angle prediction that helps in navigating the car in the right path. In this work, we aim to improve this prediction by using information from that of the lane detector. Our premise is that information from detection of lanes will be a really useful feature that can help predict the steering angle better.

1. Introduction

Over the next decade, we are about to see a huge change from manual human driving to self driving cars. This would be a great advantage to mankind if solved well as it would drastically reduce the amount of on road accidents and also improve the efficiency of the road network movement.

Udacity has an ongoing self driving car challenge with an objective to create an open source self driving car in which the second challenge was to be able to predict the car steering angle accurately. They have released a dataset of images that were taken while driving along with the corresponding steering angle. The goal of the challenge was to find a model that given a new photo to be able to predict the steering angle such that the RMSE(root mean square error).

We believe that for this purpose, detection of the lane would help in predicting better steering angles rather than simply using the image. We aim to first train a network that would be able to predict the lane detection accurately and then be able to use those output features in concatenation with the original image and pass through another network that would be able to predict the steering angle. Hypothetically, we think that this would work well because the steering angle prediction is very much based on the lane curvature of the road which can be captured by the lane detection network. We then aim to conduct experiments that will show the effectiveness or ineffectiveness of the proposed method.

2. Problem Statement

Using our methodology, we can divide the problem into two parts - one is the section to detect lanes correctly, second is to use this information to be able to predict right steering angle.

2.1. Part 1

In this part, we first aim to train a network that is highly capable of being able to detect the lane from the picture. For this, we use the dataset collected from this link : https://github.com/mvirgo/MLND-Capstone. In this dataset, there are 11487 colored images of the road with resolution of 80x160 pixels.

We expect that this network will accurately be able to detect the lanes reliably. Evaluation on this can be done by measuring the MSE loss of the pixels on the predicted output compared to the original output.

2.2. Part 2

This part deals with the main aim of predicting the steering angle of the vehicle. Our premise is that we would like to use the output of the lane detector concatenated with the input image to predict the steering angle. For this purpose, we would train the part 1 as described above and as a preprocessing step, use the parameters learned to generate lane detected outputs from input images, append it to the input and pass through another network that would predict the steering angle.

The dataset that we plan to use in this case is that of the Udacity self driving car challenge : https://github.com/udacity/self-driving-car/tree/master/datasets. We expect that using this methodology will help to improve the predictions of the steering angle. Evaluations in this case can be done by measuring the mean square error of the predicted angle.

3. Technical Approach

The problem can be divided into 2 parts. First, we train a deep learning model for predicting the lane on the roads.

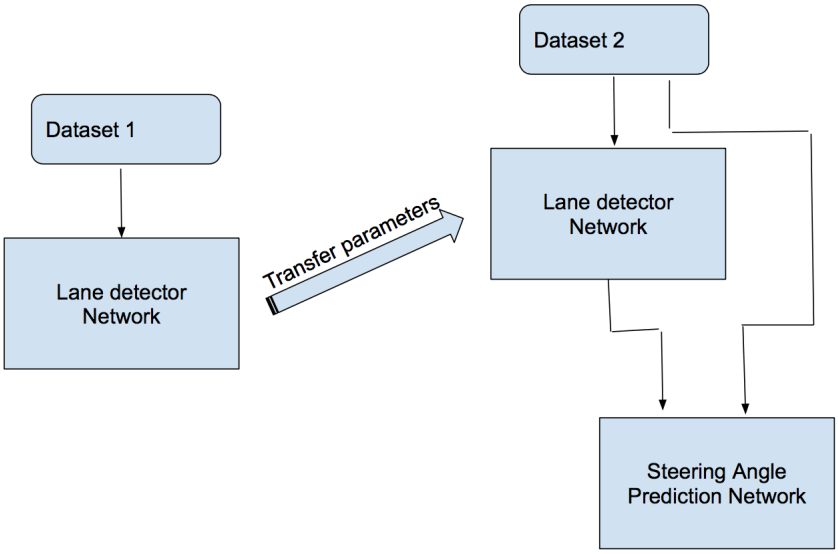

Figure 1. Proposed Model

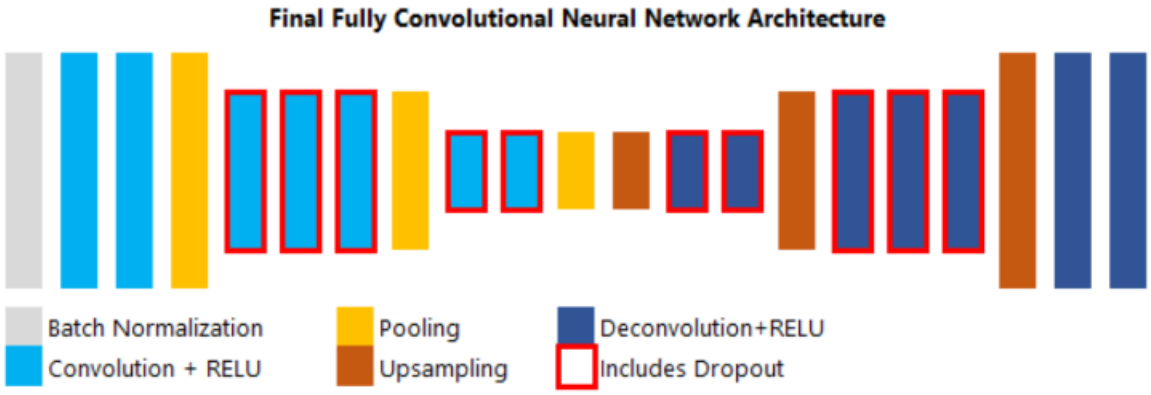

Figure 2. Lane detector CNN architecture

Then, we save the model for transfer learning. The output of this network is another image with the lane marked in 'green' color. In second step, we train a neural network for predicting steering angles on the road. It is a regression network with 2 inputs:-

1. Input image

2. Output image from lane detector network

The output of this network is a scalar, that determines how much should the steering wheel rotate. In this way, we serve an additional input of identified lane to the second network and increase the accuracy.

3.1. Lane Detector Architecture

The input images were shifted using ImageDataGenerator in Keras for data augmentation. CNN architecture (Fig.1) is followed, with max pooling (2,2). The dropout was set to 0.2.

Layer (type) Output Shape
batchnormalization2 (Batch (None, 80L, 160L, 3L)
Conv1 (Conv2D) (None, 78L, 158L, 8)
Conv2 (Conv2D) (None, 76L, 156L, 16)
maxpooling2d4 (MaxPooling2 (None, 38L, 78L, 16)
Conv3 (Conv2D) (None, 36L, 76L, 16)
dropout11 (Dropout) (None, 36L, 76L, 16)

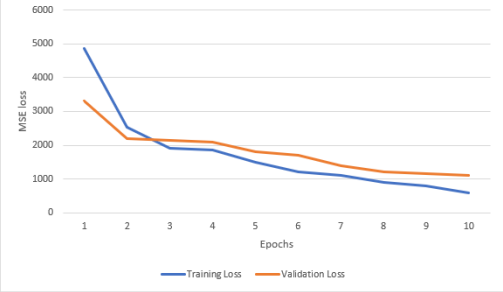

Figure 3. The loss curve for initial training

Conv4 (Conv2D) (None, 34L, 74L, 32)
dropout12 (Dropout) (None, 34L, 74L, 32)
Conv5 (Conv2D) (None, 32L, 72L, 32)
dropout13 (Dropout) (None, 32L, 72L, 32)
maxpooling2d5 (MaxPooling2 (None, 16L, 36L, 32)
Conv6 (Conv2D) (None, 14L, 34L, 64)
dropout14 (Dropout) (None, 14L, 34L, 64)
Conv7 (Conv2D) (None, 12L, 32L, 64)
dropout15 (Dropout) (None, 12L, 32L, 64)
maxpooling2d6 (MaxPooling2 (None, 6L, 16L, 64)
upsampling2d4 (UpSampling2 (None, 12L, 32L, 64)
Deconv1 (Conv2DTranspose) (None, 14L, 34L, 64)
dropout16 (Dropout) (None, 14L, 34L, 64)
Deconv2 (Conv2DTranspose) (None, 16L, 36L, 64)
dropout17 (Dropout) (None, 16L, 36L, 64)
upsampling2d5 (UpSampling2 (None, 32L, 72L, 64)
Deconv3 (Conv2DTranspose) (None, 34L, 74L, 32)
dropout18 (Dropout) (None, 34L, 74L, 32)
Deconv4 (Conv2DTranspose) (None, 36L, 76L, 32)
dropout19 (Dropout) (None, 36L, 76L, 32)
Deconv5 (Conv2DTranspose) (None, 38L, 78L, 16)
dropout20 (Dropout) (None, 38L, 78L, 16)
upsampling2d6 (UpSampling2) (None, 76L, 156L, 16)
Deconv6 (Conv2DTranspose) (None, 78L, 158L, 16)
Final (Conv2DTranspose) (None, 80L, 160L, 1)

4. Preliminary Results

The lane detector was trained on the dataset mentioned above in the part 1 of the problem statement.

We replicated the state-of-the-art lane detector using Keras library. The training took 7 hours and we tested the code successfully. The final training loss over 10 epochs was 654. In the final transfer learning, we plan to train over 20 epochs. There are 11,300 images, so final training loss per image is 0.0578.

The validation test was kept at 10 percent of training set. The final validation loss over all sample (10 epochs) was 1132.

The results of this experiment were as expected and can be seen in Figure 3.