Self Driving Car Nanodegree Project 3

Michael Person

March 12, 2017

1 Introduction

The purpose of the third project was to train a Convolutional Neural Network to steer a vehicle around a track inside an emulator. The network would read in camera's on the front of the car and then calculate the steering angle that the car needed to take. The training data was therefore an image and a numerical value for the angle making this a regression problem.

2 Data Augmentation

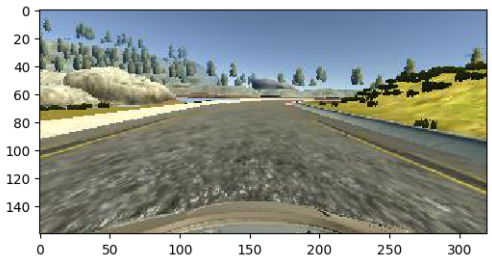
A variety of data augmentation methods were tried such as brightness alteration, horizontal shifting, and image mirroring. However the only augmentation that yielded good results on the track was image mirroring.

All three camera views were used including angle shifts in the left and right angles to account for a error correction which would allow the car to correct itself if it ever found it was off the center of the track.

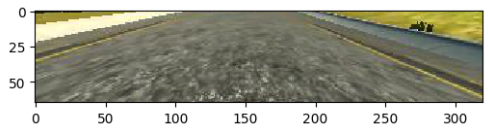
The data used was the training data that Udacity supplied. No additional data was required because Dropout was added.

In order to teach the car to drive over the shadows, the images were converted to YUV colorspace and the network was trained on these images and the drive script was also altered to accommodate for this different colorspace.

The original image becomes this:



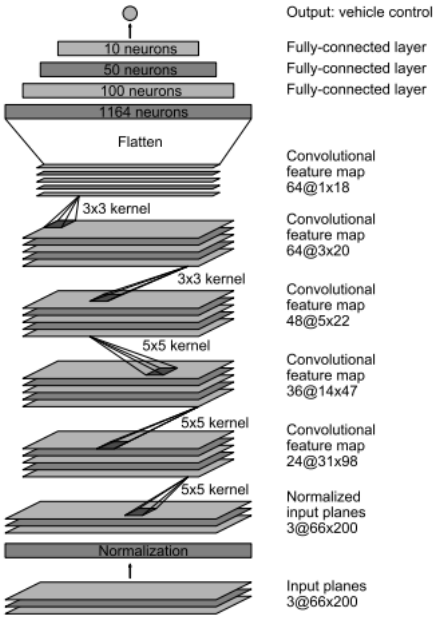
The cropped image becomes this:



3 Model Architecture

The model architecture that was used was Nvidia's End to End architecture. This model seemed appropriate for the task given it was specifically designed to drive a vehicle rather than trying to repurpose Alexnet or another common network. The only difference was that Dropout was added in after the 3rd, 5th, and 7th layers. Dropout allowed the network to train without the need of

any additional data. Max pooling was also added in after the 4th and 6th layers with kernel size 2x2 and stride of 1.

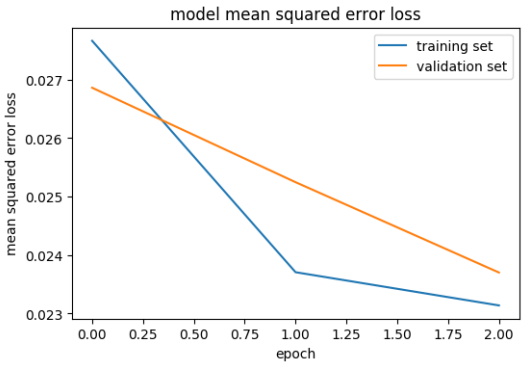


4 Training

The network was trained using the Adam optimizer because of it's efficiency with large datasets and models such as neural networks. The model was trained for only three epochs because although loss monotonically decreased, the network failed to complete the track if more training was done. A batch size of 256 was used.

5 Results

The car was able to make it around the track easily. However the car would fail miserably on either the jungle or mountain challenge tracks. I tried many many things in order to make the car work on those tracks but I made no headway and decided that it was best to quit.



I tried different models, different hyperparameters, different training datasets, different augmentations, and different colorspaces. Not only did I see marginal improvements on either of the challenge courses but I saw severe decreases in performance on the regular course. Therefore I decided that it would be best to stick with a simple, plain jane approach because 100% of something working is better than 50% of something partially working.