Behavioral Cloning

Jianxing Ke

January 1, 2019

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

1 Files Submitted

My project includes the following files:

- model.py containing the script to create and train the model
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- CarND-Behavioral-Cloning-P4-report.pdf summarizing the results

The folder '/IMG' contains training data collected from the simulator, and 'driving_log.csv' contains file names, steering angle, etc from each image. First we need to run 'python model.py' to train the model. Assume the model is named 'model.h5', then we can run 'python drive.py model.h5 run1', which will output the predicted steering angles to the simulator if it's in autonomous mode.

2 Training Data

2.1 Data Collection

The training data was collected by running the simulator, which consisted of images coming from three cameras mounted behind the windshield of the car.

One camera at the center position, one at the left and one at the right. My first attempt was that I drove the car normally for about two laps; turned around and drove one lap; several recovery runs from the sides of the road back to the center. Unfortunately this dataset didn't work well when I used it to train my model because I drove the simulated car using my keyboard instead of a joystick, and the turns were not smooth at all. The final dataset I used to train my model was the sample dataset provided.

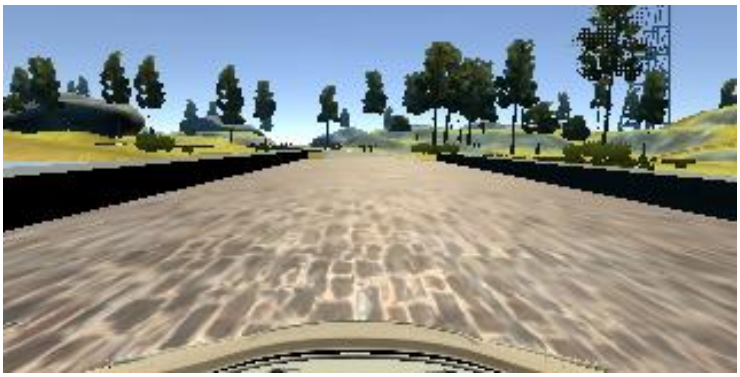


Figure 1: A center image

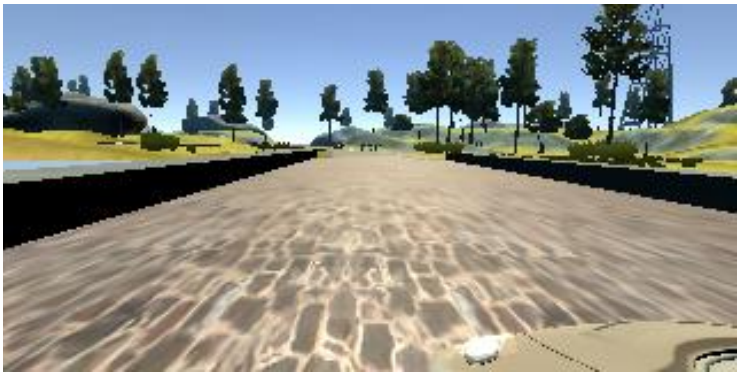


Figure 2: A left image

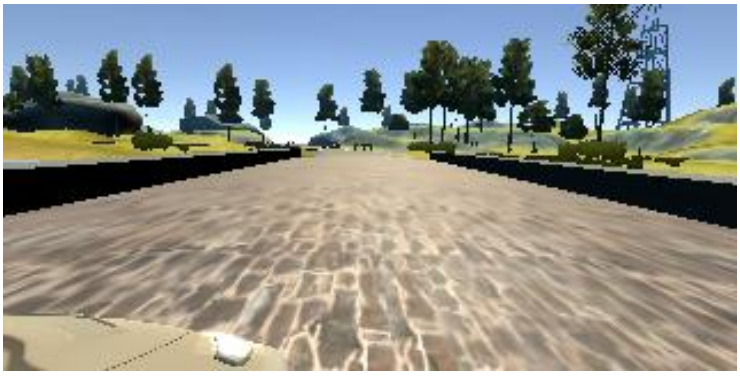


Figure 3: A right image

2.2 Data Augmentation

The first step of processing the dataset was selecting the images to use. Inspired by Nvidia's paper that we need to remove the bias towards driving straight by

including more images representing road curves, I looked at the driving_log.csv file and counted that it had 3675 frames (3 images from 3 cameras) out of 8036 frames with nonzero steering angles. A larger portion of the dataset had zero steering angles. I took all 3675 frames with nonzero steering angles, and randomly choose 50% of the frames with zero steering angles.

The next step was to generate augmented data. The first track in the simulator has a left turn bias. To remove that bias, I flipped each image left to right and negated the steering angle. For images taken from the left camera, I added an angle adjustment 0.15 so that the car would turn more to the right, and for images taken from the right camera, I subtracted the same angle adjustment to make the car turn more to the left.

Another step of data preprocessing before the model training was to crop the images. The original image size from the simulator was 160x320x3. Looking at the image, we see that the upper portion of the image contains sky, and objects such as trees and mountains, and we don't need that information. Also, the bottom part of the image is the front of our simulated car. To get rid of these two parts, I cropped the image to be 90x320x3, 50 pixels from the top of the image and 20 pixels from the bottom.

Figure 4: A cropped image

The last step was normalization so that the dataset had zero mean and the same deviation.

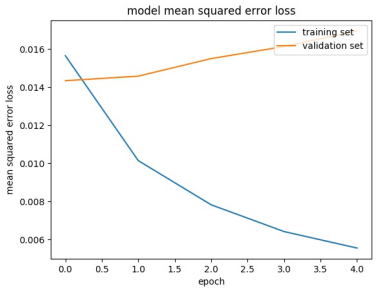
3 Model Architecture

The training model took images with size 90x320x3 as input, and output the predicted steering angle. The weights were calculated by minimizing the mean squared error between the input steering angle and the predicted angle using adam optimizer. I first started with LeNet structure, but it didn't perform well and the car went off the track soon after the simulation started. Then I tried the architecture in Nvidia's paper. With some modifications, my final model had a cropping layer, followed by a normalization layer, followed by 4 convolution layers with relu activation, each convolution layer had a dropout layer with dropped rate 0.2; then a flatten layer, followed by 3 fully connected layers, and the final output was the predicted steering angle. The dataset was split so that 80% was the training data and 20% was the validation data. The batch size was 32 and the number of epochs was 5.

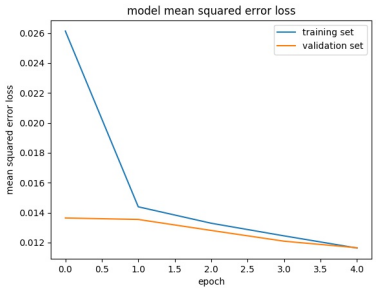
4 Results & Testing

The trained model from the above architecture was saved as 'model.h5' and the video recording 'run1.mp4'. From the video, we can see that the model performed pretty well. The simulated car stayed on track the whole time and made smooth turns.

I also plot the training errors and validation errors to see how the model performed with this dataset. The left plot was from the above model without dropout layers. Clearly the model was overfitting. The right plot showed a nicer validation error curve.



(a) Errors without Dropouts



(b) Errors with dropouts