**Behavior Cloning**

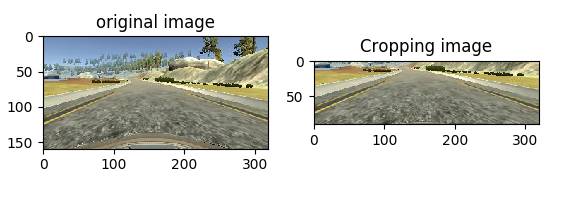
The goals / steps of this project are the following:

* 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

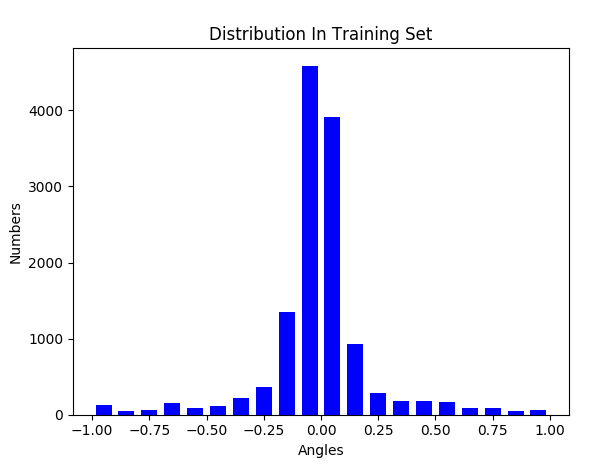
**Data Collection and Visualization**

I use the simulator to collect data of good driving behavior. First, I collect the data of five laps’ correct driving. Then I collect the data of one lap’s reverse driving the generalize the data. Finally, I collect one lap’s data of sine driving so that the car can learn to go back to the center when it is off the road.

I do cropping on the image and here is the result:



Here is the distribution of data:



**Model for training**

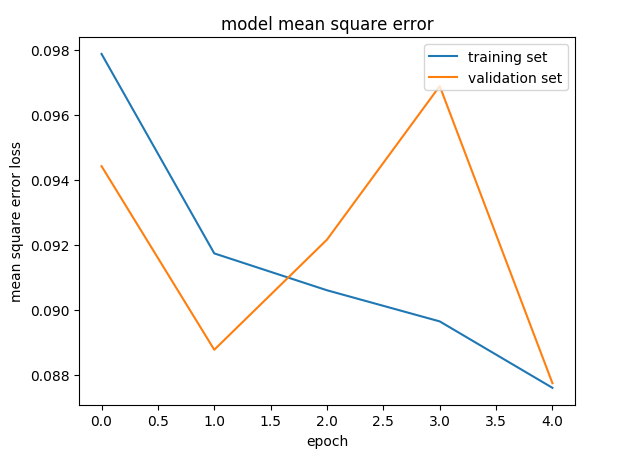
I first tried Lenet for training a model. However, after times of trying, I found that the performance was not good. So, finding on the Internet, I use the NVIDIA’s model structure. Since it has five convolutional layers, it can capture many details, and since NVIDIA uses this model to train real car, I think it definitely has the ability to train my car. Based on that, I changed it a little bit by adding **Dropout** layers to prevent the overfitting. Here is the visualization of my model structure. By the way, I use the **Adam optimizer**.

|  |  |
| --- | --- |
| Layer | Output |
| / | 160 \* 320 \* 3 |
| Cropping2D | 90 \* 320 \* 3 |
| Convolutional2D, ‘relu’, Dropout | 43 \* 158 \* 24 |
| Convolutional2D, ‘relu’, Dropout | 20 \* 77 \* 36 |
| Convolutional2D, ‘relu’, Dropout | 8 \* 37 \* 48 |
| Convolutional2D, ‘relu’, Dropout | 3 \* 18 \* 64 |
| Convolutional2D, ‘relu’, Dropout | 1 \* 16 \* 64 |
| Flatten | 1046 |
| Dense | 100 |
| Dense | 50 |
| Dense | 10 |
| Dense | 1 |

**Training Set and Validation Set**

I use the generator function provided on Udacity’s course. I also do a little change on it to augment data. I randomly choose the camera image. If it is the right camera, the angel will minus -0.25, if it is the left camera, the angle will add 0.25. Also, I flip the image horizontally to generalize the data.

Here is the loss:

****

I also tried for 10 epochs, but the performance is worse, so I chose to train for only five epochs.

**Test Video**

See “Behavior Cloning.mp4”