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
- `writeup_report.pdf` summarizing the results

I used the Nvidia end-to-end Convolutional Neural Network architecture, which consists of 9 layers, including a normalization layer, 5 convolutional layers, and 3 fully connected layers. In addition, I added a cropping layer to get rid of the sky and hood in the images, as well as 3 dropout layer between convolution layers and 2 dropout layers between fully connected layers to suppress the over-fitting. The activation algorithm is RELU.

The model used an adam optimizer, and loss function is *mean squared error* (*MSE*).

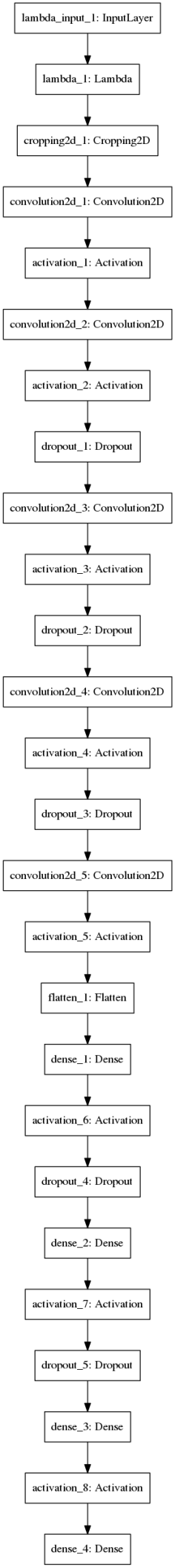
I collected about 7k images data by driving the car in the simulator. I drove clockwise and counterclockwise on track 1, as well as many recovery recording.

I combined my data with the Udacity training data, then shuffled and split them into training set and validation set with roughly 80/20 ratio by using linux shell.

Different batch size (64, 32, and 28) has been tried, it seems that 32 works the best in my case. I have about 16k images in training set, considering the augmentation, 20k to 30k samples per epoch seems reasonable. I found a student used 23296 samples_per_epoch, and it compatible with batch size of 32, so I used this number, and it worked well. I tried 5 epochs and 10 epochs, 10 epochs seems to make the car driving more smoothly.

The biggest problem of the training set is that it is unbalanced, the majority of the sheering angle is 0 or very close to 0. To tackle the issue, I setup a filter to filter out part of this group. The result is a faster converge.

Originally, I just used the center camera images of Udacity data set and two drop-out layers in my model. After trained the model, the car in autonomous model can keep in the track, buy running likes a drunk driver was in the driver seat. I added more training data by driving on the track in both direction to avoid too many left turn, which will help the model generalize better. The result is more smooth driving experience.


lambda_input_1: InputLayer
lambda_1: Lambda
cropping2d_1: Cropping2D
convolution2d_1: Convolution2D
activation_1: Activation
convolution2d_2: Convolution2D
activation_2: Activation
dropout_1: Dropout
convolution2d_3: Convolution2D
activation_3: Activation
dropout_2: Dropout
convolution2d_4: Convolution2D
activation_4: Activation
dropout_3: Dropout
convolution2d_5: Convolution2D
activation_5: Activation
flatten_1: Flatten
dense_1: Dense
activation_6: Activation
dropout_4: Dropout
dense_2: Dense
activation_7: Activation
dropout_5: Dropout
dense_3: Dense
activation_8: Activation
dense_4: Dense