Model Architecture and Training Strategy

1.Model Architecture:

For LeNet architecture car was drifting off the road, so I tried more powerful architecture from NVIDIA end to end driving paper. In my model I used 3 convolution layers with 5x5 kernel size followed by 3 more convolution layer with 3x3 kernel size followed by 4 fully connected layers. The last fully connected layer is of size 1 to predict steering angle.
At beginning data is normalized in the model using a Keras lambda layer and cropped using keras cropped layer. The model includes RELU layers to introduce nonlinearity.

2. Attempts to reduce over fitting in the model:

First training on above model showed very high mean squared error(mse), loss between training and validation set resulted in car drifting off the road coz of over fitting. To reduce over fitting, I increased number of data points and introduced dropout and pooling layers in model which. This strategy resulted in reducing mse loss at the end car successfully completed lap.

3. Model parameter tuning

The model used an adam optimizer, so the learning rate was not tuned manually.

4. Appropriate training data

I used all 3 camera images to train my model. I changed steering angle correction factor from 0.2 to 0.4 which showed lot improvement in driving. While recording data I tried to keep car in middle of road. I let the car drift to the edge of the road and recover before a crash occurs. I took 2 laps of center lane and one lap of track from opposite direction. I got around 4825 data points for training my model.

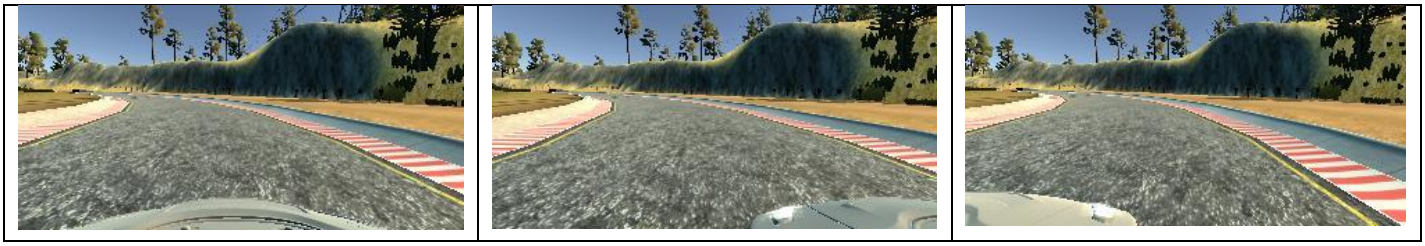

Figure1: Camera Images from center, left and right

5. Solution Design Approach
In designing model architecture I was looking to minimize over fitting and converging solution as fast as possible on given data. My first approach was to use simple model (LeNet) and then I used more powerful model from NVIDIA paper which showed a lot of improvement compared to earlier architecture, after minimizing over fitting and lot of tweaks my car able to complete track.

6. Final Model Architecture and analysis.

Following is final model architecture.

Layer (type)	Output Shape	Param #
cropping2d_1 (Cropping2D)	(None, 65, 320, 3)	0
lambda_1 (Lambda)	(None, 65, 320, 3)	0
conv2d_1 (Conv2D)	(None, 33, 160, 24)	1824
conv2d_2 (Conv2D)	(None, 17, 80, 36)	21636
max_pooling2d_1 (MaxPooling2	(None, 8, 40, 36)	0
dropout_1 (Dropout)	(None, 8, 40, 36)	0
conv2d_3 (Conv2D)	(None, 4, 20, 48)	43248
max_pooling2d_2 (MaxPooling2	(None, 2, 10, 48)	0
dropout_2 (Dropout)	(None, 2, 10, 48)	0
conv2d_4 (Conv2D)	(None, 2, 10, 64)	27712
conv2d_5 (Conv2D)	(None, 2, 10, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 1, 5, 64)	0
dropout_3 (Dropout)	(None, 1, 5, 64)	0
flatten_1 (Flatten)	(None, 320)	0
dense_1 (Dense)	(None, 100)	32100
dropout_4 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 50)	5050
dropout_5 (Dropout)	(None, 50)	0
dense_3 (Dense)	(None, 10)	510
dense_4 (Dense)	(None, 1)	11

Total params: 169,019.0
Trainable params: 169,019.0
Non-trainable params: 0.0

For above architecture at first I used 2048 data points for training which resulted in over fitting as shown in figure 2, to minimize over fitting I captured more training data and change in strategy of data logging explained in 'Appropriate training data' section. For 4825 data points I got result shown in figure 3.

.

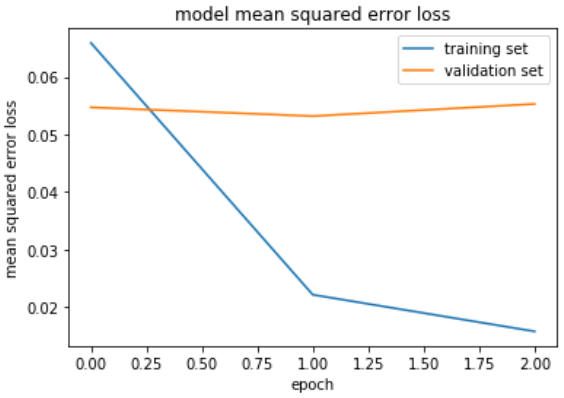

Figure 2: MSE with small data

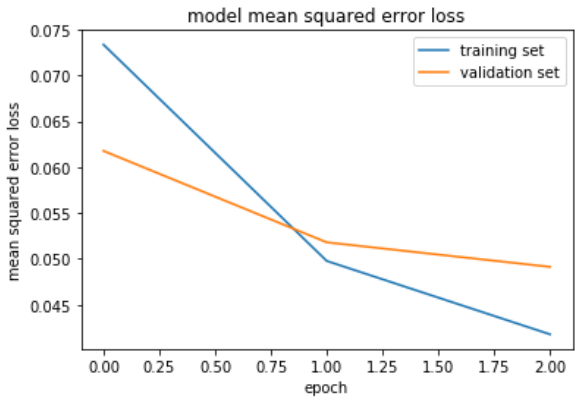

Figure 3: MSE with Large data (Final solution)