I think it is better to fix the problem with provided data path, there is a space key in front left and right path.

Index	Type	Size	Value
0	str	1	IMG/center_2016_12_01_13_30_48_287.jpg
1	str	1	IMG/left_2016_12_01_13_30_48_287.jpg
2	str	1	IMG/right_2016_12_01_13_30_48_287.jpg
3	str	1	0
4	str	1	0
5	str	1	0
6	str	1	22.14829

Behavioral Cloning

Writeup Template

Behavioral Cloning Project

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

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Files Submitted & Code Quality

1. Submission includes all required files and can be used to run the simulator in autonomous mode

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.md or writeup_report.pdf summarizing the results

2. Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously around the track by executing

python drive.py model.h5

3. Submission code is usable and readable

The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

Model Architecture and Training Strategy

1. An appropriate model architecture has been employed

The Nvidia Model (without 1064 FC)suggested on the course was used.

The model includes RELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras lambda layer.

2. Attempts to reduce overfitting in the model

Dropout layers are not used in this task.

The model was trained and validated on different data sets to ensure that the model was not overfitting. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

3. Model parameter tuning

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

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road

For details about how I created the training data, see the next section.

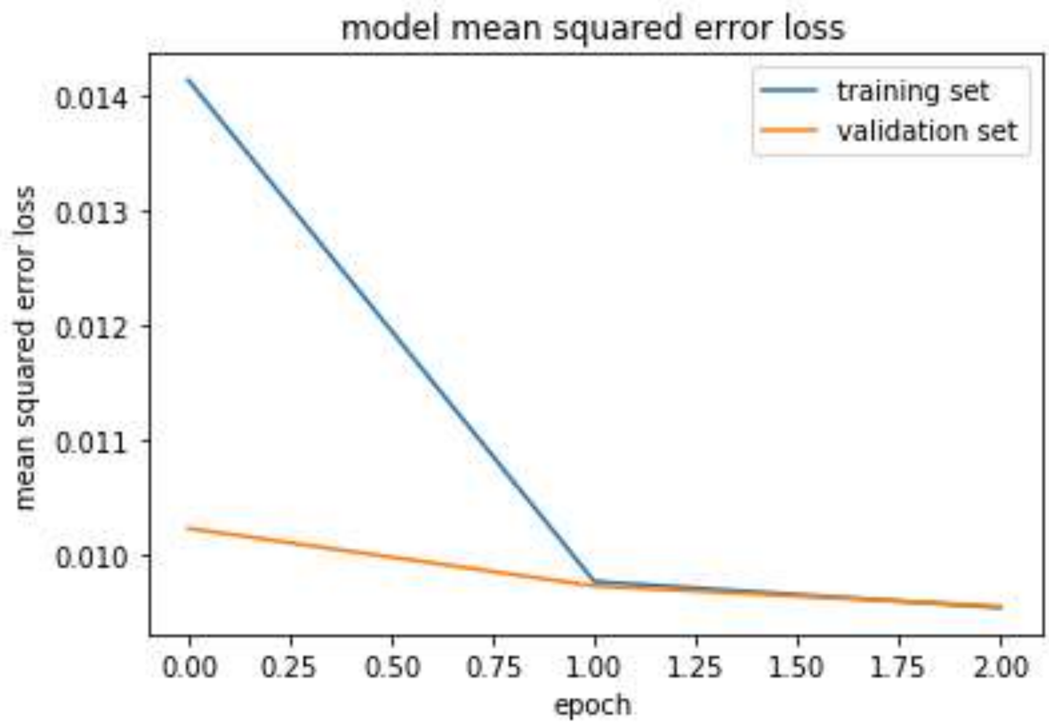
Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to use a fully connected network with data from center camera to ensure the pipeline working

Then I use a convolution neural network model similar to the Nvidia Model (without 1064 FC) suggested on the course with center images. I did Normalization with lambda and add a cropping layer which removed the top 60 and bottom 20 pixels from each input image before passing the image on to the convolution layers.

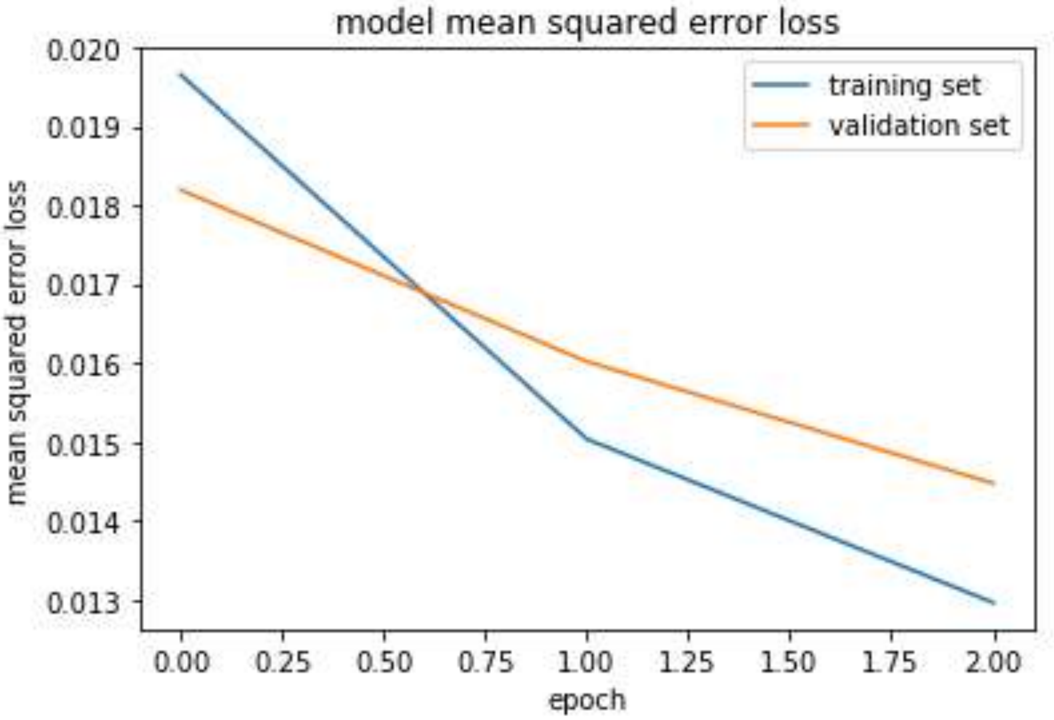
Here is the result for MSE. The MSE for the training set and validation set is close, so I do not see a overfitting problem. So I don’t need to deal with overfitting.



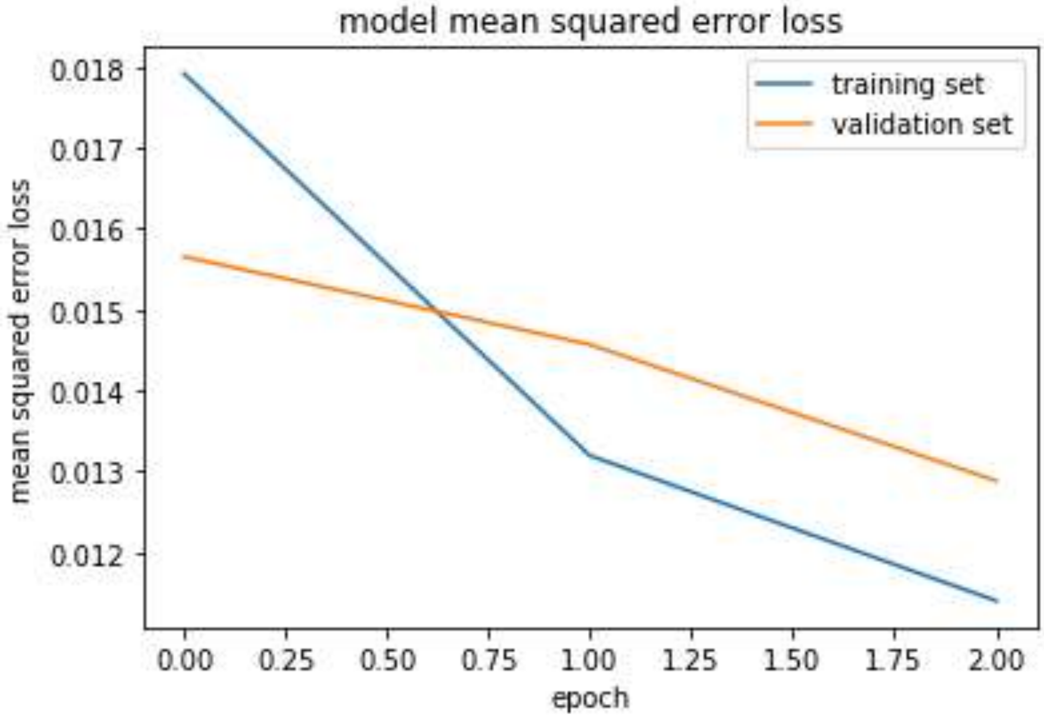
I also check the simulation result which is not that good. The vehicle has run off the road after passing the ridge in a turn.

I think there are two ways to improve this model. One is to collect images when the car is driving from the side of the road back toward the center line. So the model can learn how to drive back to the center. Another way is to use more data to teach vehicle keep in the center.

The second way is easy to do so I add left and right camera data to train the model. Here is the result for MSE. It seems the MSE increases, but it makes sense since I use more data and the correction of left and right measurements is simply chosen as 0.2. I expect it can run much better.



However it doesn’t run as smoothly as the first one. I try to modified the network, I plan to add a maxpooling layer before flatten, to reduce the parameters in the first FC layers.(from 6336->100 to 2048-)100) I hope the training can be faster and the results becomes better.



At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

Version 1: reduce 1064FC

Layer (type)	Output Shape	Param #
lambda_5 (Lambda)	(None, 160, 320, 3)	0
cropping2d_5 (Cropping2D)	(None, 80, 320, 3)	0
conv2d_21 (Conv2D)	(None, 38, 158, 24)	1824
conv2d_22 (Conv2D)	(None, 17, 77, 36)	21636
conv2d_23 (Conv2D)	(None, 7, 37, 48)	43248
conv2d_24 (Conv2D)	(None, 5, 35, 64)	27712
conv2d_25 (Conv2D)	(None, 3, 33, 64)	36928
flatten_5 (Flatten)	(None, 6336)	0
dense_17 (Dense)	(None, 100)	633700
dense_18 (Dense)	(None, 50)	5050
dense_19 (Dense)	(None, 10)	510
dense_20 (Dense)	(None, 1)	11

Total params: 770,619
Trainable params: 770,619

Version 2:add maxpooling Fewer parameters, better result

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 160, 320, 3)	0
cropping2d_1 (Cropping2D)	(None, 90, 320, 3)	0
conv2d_1 (Conv2D)	(None, 43, 158, 24)	1824
conv2d_2 (Conv2D)	(None, 20, 77, 36)	21636
conv2d_3 (Conv2D)	(None, 8, 37, 48)	43248
conv2d_4 (Conv2D)	(None, 6, 35, 64)	27712
conv2d_5 (Conv2D)	(None, 4, 33, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 2, 16, 64)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 100)	204900
dense_2 (Dense)	(None, 50)	5050
dense_3 (Dense)	(None, 10)	510
dense_4 (Dense)	(None, 1)	11

Total params: 341,819
Trainable params: 341,819

3. Creation of the Training Set & Training Process

Only use the given sample data and follow some tips like image cropping and data normalization. I would like to find an excuse: I am so lazy that I don't even want to play games.