Behavioral Cloning

Writeup

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 (https://review.udacity.com/#!/rubrics/432/view) 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
- train_network.ipynb: the original ipython notebook (model.py is exported from this file)
- drive.py for driving the car in autonomous mode
- model.h5 containing a trained convolution neural network
- writeup.pdf (and writeup.ipynb) 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

I've tried three different model architectures:

- a model with only Conv2D and Dense layers
- a model using Dropout to avoid overfitting.
- a model using Dropout and Max Pooling to avoid overfitting

1. An appropriate model architecture has been employed

Model 1: a model using only Conv2D and Dense layers

- When training only with Conv2D and Dense, the model was trained rapidly and the loss has reduced quickly.
- However, the validation loss didn't drop and was oscillating within around 0.04 of validation loss.
- It means the model is suffering from overfitting.

The following tables shows the model architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 160, 320, 3)       0
_________________________________________________________________
cropping2d_1 (Cropping2D)    (None, 65, 320, 3)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 31, 158, 24)       1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 77, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 37, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 35, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 33, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 2112)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               211300
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 348,219
Trainable params: 348,219
Non-trainable params: 0
```

As you can see in the graph below, the loss values decrease quite fast and well enough (0.0039), but the validation loss didn't decreased, but diverged after 4 epochs.

The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track. The vehicle was successfully on the track for two laps.

2. Attempts to reduce overfitting in the model

Model 2: a model with Dropout

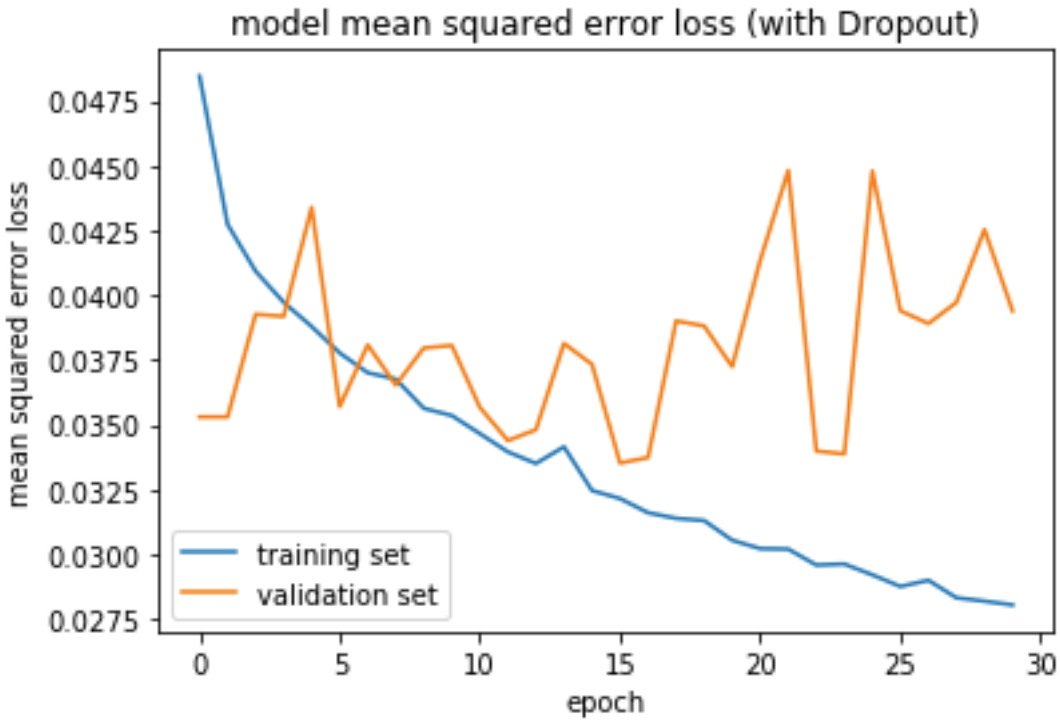
- To fix the overfitting, we tried to apply Dropout layers on several layers.
- With Dropout, I had to increase the epoch count because the loss value was reduced much slower than when trained without Dropout.
- The validation loss seemed decreasing at first few epochs, but after that it diverged again
- I applied Dropout only three times for convolutional network, and two times for fully connected networks.
    - When I tried to apply Dropout to every layer, the model was trained very slowly, and also the loss didn't reduce much.

The following tables shows the model architecture:

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 160, 320, 3)	0
cropping2d_1 (Cropping2D)	(None, 65, 320, 3)	0
conv2d_1 (Conv2D)	(None, 31, 158, 24)	1824
dropout_1 (Dropout)	(None, 31, 158, 24)	0
conv2d_2 (Conv2D)	(None, 14, 77, 36)	21636
conv2d_3 (Conv2D)	(None, 5, 37, 48)	43248
dropout_2 (Dropout)	(None, 5, 37, 48)	0
conv2d_4 (Conv2D)	(None, 3, 35, 64)	27712
conv2d_5 (Conv2D)	(None, 1, 33, 64)	36928
dropout_3 (Dropout)	(None, 1, 33, 64)	0
flatten_1 (Flatten)	(None, 2112)	0
dense_1 (Dense)	(None, 100)	211300
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: 348,219
Trainable params: 348,219
Non-trainable params: 0

As you can see in the graph below, the loss values decrease not that fast as the previous model did and didn't get good loss values within 30 epochs. The average of the validation loss was a little smaller than the model1 lodecreased, but diverged badly after 15 epochs.



The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track. The vehicle was successfully on the track for two laps, althouth it hit the guardrail on the right on the bridge.

Model 3: a model with Dropout and Pooling

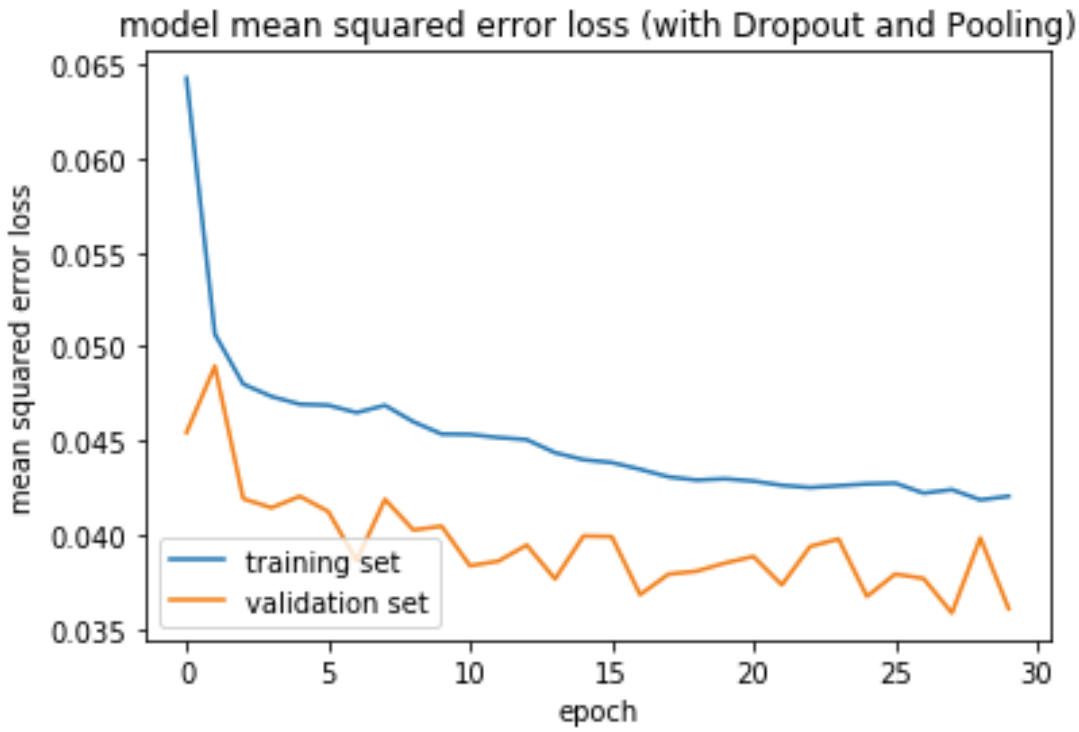
- Another try to fix the overfitting was to add Pooling layers on several layers.
- To be able to apply Pooling layers, we added a Conv2D layer with (3, (1, 1)) at the front of the model to represent color space.

The following tables shows the model architecture:

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 160, 320, 3)	0
cropping2d_1 (Cropping2D)	(None, 65, 320, 3)	0
conv2d_1 (Conv2D)	(None, 65, 320, 3)	12
conv2d_2 (Conv2D)	(None, 63, 318, 16)	448
dropout_1 (Dropout)	(None, 63, 318, 16)	0
conv2d_3 (Conv2D)	(None, 30, 157, 32)	12832
max_pooling2d_1 (MaxPooling2	(None, 15, 78, 32)	0
dropout_2 (Dropout)	(None, 15, 78, 32)	0
conv2d_4 (Conv2D)	(None, 7, 38, 48)	13872
max_pooling2d_2 (MaxPooling2	(None, 3, 19, 48)	0
dropout_3 (Dropout)	(None, 3, 19, 48)	0
conv2d_5 (Conv2D)	(None, 1, 17, 64)	27712
dropout_4 (Dropout)	(None, 1, 17, 64)	0
flatten_1 (Flatten)	(None, 1088)	0
dense_1 (Dense)	(None, 256)	278784
dropout_5 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_6 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
dropout_7 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 8)	520
dense_5 (Dense)	(None, 1)	9

Total params: 375,341
Trainable params: 375,341
Non-trainable params: 0

As you can see in the graph below, the validation loss seemed lower than the previous models, but the overall loss value is higher and not decreasing fast enough. Even with epochs=50, the loss remained as 0.0421, which is a lot higher than other previous models.



The model was tested by running it through the simulator, and the vehicle could not successfully stay on the track. I think it was that the model was not trained well enough and it ended with the loss of 0.0421 which is relatively high.

3. Model parameter tuning

- The model used an adam optimizer, so the learning rate was not tuned manually.
- epochs = 30 for the model1 and the model2, and 50 for the model3.

4.Creation of the Training Set and Appropriate training data

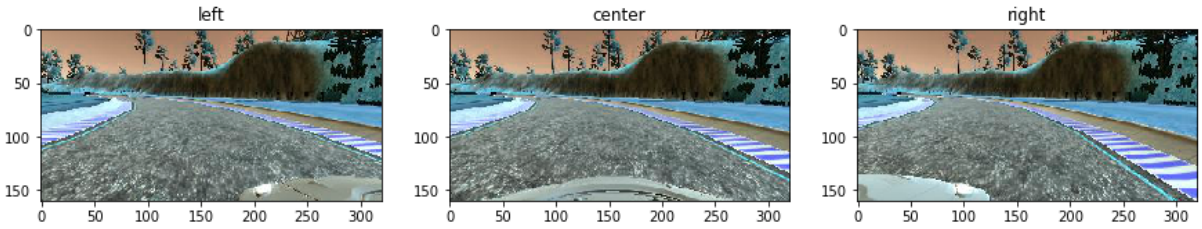
I used three data (data1, data2 and data3) for training.

- data1: the data captured from the simulator while I tried to drive the car in the center of the lane. The data were generated for two laps.
- data2: the data captured from the simulator while I tried to drive the car in recovery driving, that is, I tried to move the car to the edge of each side of the lane and then quickly move back to the center, and keep doing that repeatedly. The data is for two laps.
- data3: the data captured from the simulator while I drived the car in reverse way (counter clockwise). The data is for two laps.

The input images are normalized to -0.5 to 0.5 by using Lamda function in Keras.

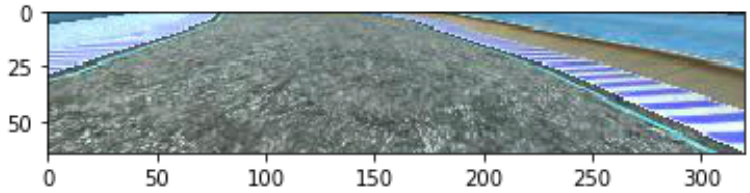
```
model.add(Lambda(lambda x: x / 255.0 - 0.5, input_shape=(160, 320,
 3)))
```

Using the images from the left and the right cameras I used three images per every shot - the images captured by the center, the left and the right cameras. For the images from the left and the right cameras, we adjusted the camera angle by a correction factor (0.2).



Cropping

I cropped the top portion (70 pixcels) of the image to remove the distracting background, and also bottom portion (25 pixels) of the images to remove the hood part of the car. The resulting image is shown below.



Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to start with a simple Convolution network, and then apply additional layers for reducing overfitting if needed.

However, when I applied the Dropout, the average validation loss seemed a little lowered, but the traning time took a lot much longer, and also the resulting validation loss was not improved much.

When I added pooling layers, the model could not trained well enough, meaning the model was training a lot much slower and the validation loss was not improved.

I have tried several other configurations of layers, and also turn on and off the pooling and dropout (with several different dropout rate), but I could not successfully improve the validation loss.

2. Final Model Architecture

The final model architecture I used for the simulator is the Model1 only with Conv2D and Dense layers, without Dropout and Pooling, because the model showed the best loss value, and the validation loss was not that bad compared to other models.

3. Creation of the Training Set & Training Process

Please see the description above.

In [ ]: