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 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

The project code and all other relevant artifacts can be verified under the following link:

https://github.com/SDCND/CarND-Behavioral-Cloning-P3

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 1 by executing

```
python drive.py model.h5
```

My model finishes track 1 without going off: TRACK 1

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

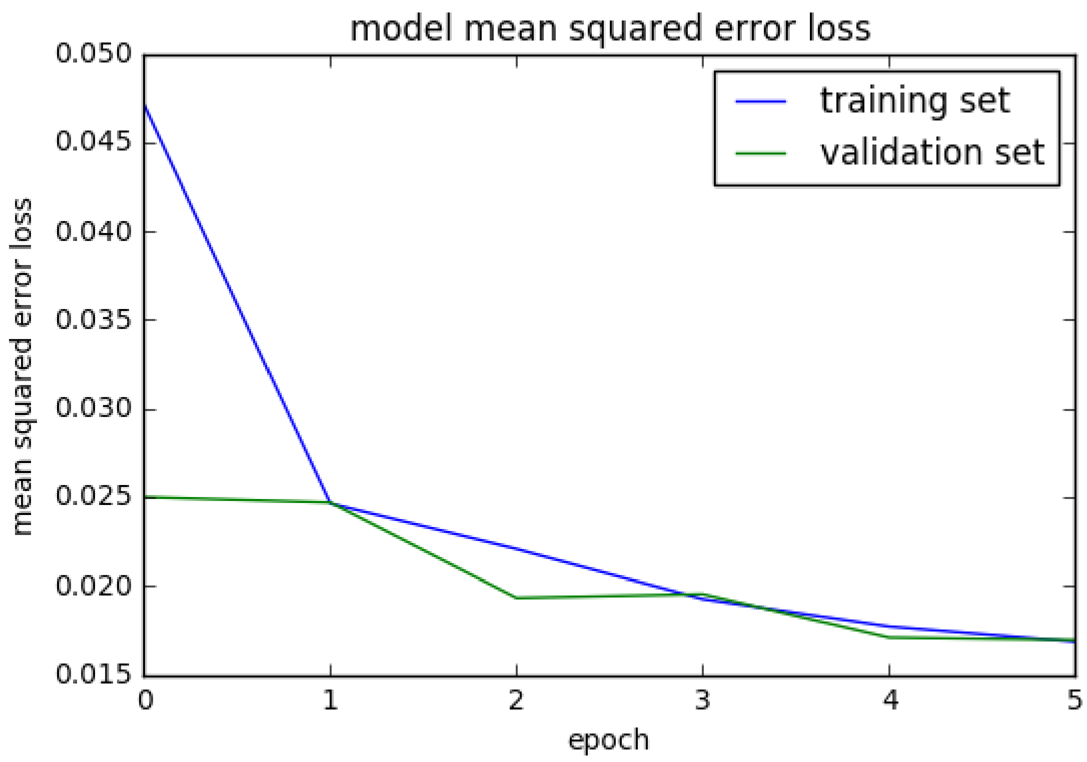
1. An appropriate model architecture has been employed

My model consists of a convolution neural network based on NVIDIA's End to End Learning for Self-Driving Cars paper. The main difference between the implemented model and the NVIDIA mode is that in this model there are used MaxPooling layers just after each Convolutional Layer to reduce the training time (model.py lines 97-142).

The model includes RELU layers to introduce nonlinearity after each Convolution2D, and the data is normalized in the model using the Keras layer BatchNormalization (code line 103).

2. Attempts to reduce overfitting in the model

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 (model.py line 144).

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.

Model Architecture and Training Strategy

1. Solution Design Approach

My first step was to use a convolution neural network model similar to the model architecture based on [Nvidia's paper](). I thought this model might be appropriate network consists of five convolutional layers, followed by three fully connected layers. It includes RELU layers to introduce nonlinearity.

During the data preprocessing there is applied a random shearing operation with a probability of 90%. There also were cropped out just the relevant image segment (without sky and car parts). With the probability of 50% there were flipped the images, because left turning bends were more relevant than right bends. Beside that images were scaled to the dimension of 64 x 64 pixels, to ensure a performant model.

2. Final Model Architecture

The final model architecture (model.py lines 97-142) consisted of a convolution neural network with the following layers and layer sizes:

```
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
batchnormalization_1 (BatchNorma (None, 64, 64, 3)     256         batchnormalization_input_1[0][0]
____________________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 32, 32, 24)    1824        batchnormalization_1[0][0]
____________________________________________________________________________________________________
activation_1 (Activation)        (None, 32, 32, 24)    0           convolution2d_1[0][0]
____________________________________________________________________________________________________
maxpooling2d_1 (MaxPooling2D)    (None, 31, 31, 24)    0           activation_1[0][0]
____________________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 16, 16, 36)    21636       maxpooling2d_1[0][0]
____________________________________________________________________________________________________
activation_2 (Activation)        (None, 16, 16, 36)    0           convolution2d_2[0][0]
____________________________________________________________________________________________________
maxpooling2d_2 (MaxPooling2D)    (None, 15, 15, 36)    0           activation_2[0][0]
____________________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 8, 8, 48)      43248       maxpooling2d_2[0][0]
____________________________________________________________________________________________________
activation_3 (Activation)        (None, 8, 8, 48)      0           convolution2d_3[0][0]
____________________________________________________________________________________________________
maxpooling2d_3 (MaxPooling2D)    (None, 7, 7, 48)      0           activation_3[0][0]
____________________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 7, 7, 64)      27712       maxpooling2d_3[0][0]
____________________________________________________________________________________________________
activation_4 (Activation)        (None, 7, 7, 64)      0           convolution2d_4[0][0]
____________________________________________________________________________________________________
maxpooling2d_4 (MaxPooling2D)    (None, 6, 6, 64)      0           activation_4[0][0]
____________________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 6, 6, 64)      36928       maxpooling2d_4[0][0]
____________________________________________________________________________________________________
activation_5 (Activation)        (None, 6, 6, 64)      0           convolution2d_5[0][0]
____________________________________________________________________________________________________
maxpooling2d_5 (MaxPooling2D)    (None, 5, 5, 64)      0           activation_5[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 1600)          0           maxpooling2d_5[0][0]
____________________________________________________________________________________________________
dense_1 (Dense)                  (None, 1164)          1863564     flatten_1[0][0]
____________________________________________________________________________________________________
activation_6 (Activation)        (None, 1164)          0           dense_1[0][0]
____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 100)           116500      activation_6[0][0]
____________________________________________________________________________________________________
activation_7 (Activation)        (None, 100)           0           dense_2[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 50)            5050        activation_7[0][0]
____________________________________________________________________________________________________
activation_8 (Activation)        (None, 50)            0           dense_3[0][0]
____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 10)            510         activation_8[0][0]
____________________________________________________________________________________________________
activation_9 (Activation)        (None, 10)            0           dense_4[0][0]
____________________________________________________________________________________________________
dense_5 (Dense)                  (None, 1)             11          activation_9[0][0]
====================================================================================================
```