Behavioral Cloning

Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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

My model consists of a convolution neural network with 5x5 filter sizes and depths between 32 and 128 (model.py lines 81-84)

The model includes RELU layers to introduce nonlinearity (code line 81,84), and the data is normalized in the model using a Keras lambda layer (code line 79).

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 83,86,90).

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 93).

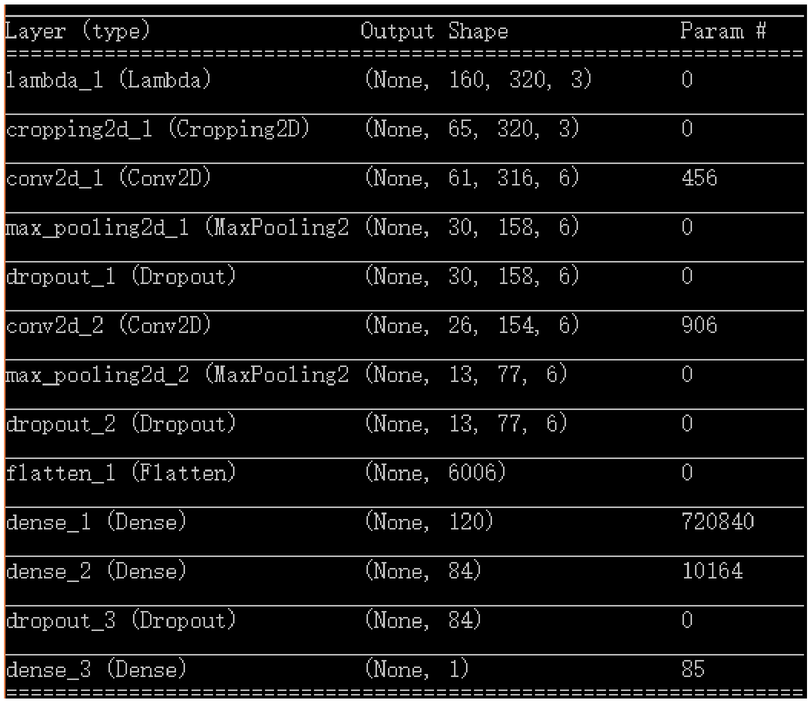
4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a simple data provided by Udacity.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was the following:



```
_________________________________________________________________
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, 61, 316, 6)        456
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 158, 6)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 158, 6)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 154, 6)        906
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 13, 77, 6)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 13, 77, 6)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6006)              0
_________________________________________________________________
dense_1 (Dense)              (None, 120)               720840
_________________________________________________________________
dense_2 (Dense)              (None, 84)                10164
_________________________________________________________________
dropout_3 (Dropout)          (None, 84)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 85
=================================================================
```

The final step was to run the simulator to see how well the car was driving around track one. It was running perfectly which shows in video.mp4.

The vehicle is able to drive autonomously around the track without leaving the road.

2. Creation of the Training Set & Training Process

To augment the data sat, I also flipped images and angles.

After the collection process, I had about 19000+ number of data points. I then preprocessed this data by normalization and cropping (model.py lines 79-80)

I finally randomly shuffled the data set and put 20% of the data into a validation set.

I used this training data for training the model. The validation set helped determine if the model was over or under fitting. The ideal number of epochs was 3. I used an adam optimizer so that manually training the learning rate wasn't necessary.