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 takes in the input and cropped them from the top and bottom 50 and 20 pixels sperately, to exclude the space above the road and the hood of the car.

Afterwards, the images are normalized by a BatchNormalization layer.

The convolutional part are all with 3x3 filter sizes and depths between 24 and 64. The first two convolution layers used 2x2 strides. In brief, it has 24 x 2, 36 x 1 and 48 x 1, and 64 x 2 layers.

Followed by a flatten layer, and fully connected layer with the width of 800-300-50-10-1. We expect one output in this task, the steering angle, so the output is 1.

The model includes RELU layers after every concolutional and fully connected layer to introduce nonlinearity.

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.

Actuallly, adding dropout layers into this model made it worse in performance. I tried adding dropout layers only after the fully connected layer; after all convolutional and fully connected layers; after all fully connected layer and after the last convolutional layer or in the middle of the convolutional layers. All this and tunning of the keep probobalities from 0.25 to 0.9, and different dropout rate at different locations, all made it worse. The car either shifting on the road, driving off the road at the turns, or made the first part turns successfully but driving off the road at the right turn.

After removing of the dropout, the model worked.

The current model seems robust enough on the track one, with both validation and test accuracys dropping after each epoch. And the scores are close at the end.

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 the provided data set.

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to driving the car in autonomous mode.

My first step was to use a convolution neural network model similar to the VVG16 I thought this model might be appropriate because it was the best model in image classification competition on ImageNet in 2014.

Anyway, the model was too big to run on my pc.

Next, I reduced the depth according the Nvidia slef-driving model with one more convolutional layer at 24 depth.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. The model was trained for 3 epochs, with both train and validation accurays dropping, without signs of over or underfitting.

Like I mentioned at 2, dropout layer is not good in this model. They were removed in the model at the end.

The final step was to run the simulator to see how well the car was driving around track one. At the end of the process, the vehicle is able to drive autonomously around the track without leaving the road.

2. Final Model Architecture

The final model architecture consisted of a convolution neural network with the following layers and layer sizes:

Layer	Description
Input	160,320,3 colour image
Cropping	50,20, 0,0, outputs 90,320,3 image
BatchNorm	
Convolution 3x3	2x2 stride, outputs 44x159x24
RELU	
Convolution 3x3	2x2 stride, outputs 21x78x24
RELU	
Max pooling 2x2	2x2 stride, outputs 10x38x24
Convolution 3x3	1x1 stride, outputs 8x36x36
RELU	
Convolution 3x3	1x1 stride, outputs 6x34x48
RELU	
Convolution 3x3	1x1 stride, outputs 4x32x64
RELU	
Convolution 3x3	1x1 stride, outputs 2x30x64
RELU	
Fully connected	input 3840 output 800
RELU	
Fully connected	input 800 output 300
RELU	
Fully connected	input 300 output 50
RELU	
Fully connected	input 50 output 10
RELU	
Fully connected	input 50 output 1

3. Creation of the Training Set & Training Process

In the generator, the images were horizontally flipped, the steering angle was also flipped, meaning the negative number of the steering angle.

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 as evidenced by increasing of validation loss after epoch 3. I used an adam optimizer so that manually training the learning rate wasn't necessary.