
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 3x3 & 5x5 filter sizes and depths between 24, 48 and 64 (model.py lines 80-88)

The model includes RELU layers to introduce nonlinearity (code line 81,90 etc), and the data is normalized in the model using a Keras lambda layer (code line 78).

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 89,95,97).

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 101).

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.

I combine udacity and my data to train the model. I recorded extra data where my car is going to off the road. That’s help to improve the model.

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

Model Architecture and Training Strategy

1. Solution Design Approach

First step is, I try all the method in the given lecture.

Next I come to Nvidia architecture but I am unable to train the model because out of memory error. Then I try to remove some of convolution layer and add the MaxPooling layer (it's decrease the size of model) in the Nvidia architecture. But again, model is not doing well. I found that my model had a low mean squared error on the training set but a high mean squared error on the validation set. This implied that the model was overfitting. Then I try to add some DropOut Layer to avoid overfitting.

The final step was to run the simulator to see how well the car was driving around track one. There were a few spots where the vehicle fell off the track, to improve the driving behavior in these cases, I generate more data at the particular case and train with the data.

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 (model.py lines 18-24) consisted of a convolution neural network with the following layers and layer sizes ...

Cropping2D	(70,25),(0,0) input_shape = (160,320,3))
Lambda	(lambda X: X/127.5 - 1)
Convolution Activation MaxPooling	24 5x5 Relu Keras define it
Convolution Activation MaxPooling	48 5x5 Relu
Convolution Activation Dropout MaxPooling	64 3x3 Relu 0.5

Flatten	
Dense Activation Dropout	100 Relu 0.5
Dense Activation	50 Relu
Dropout	0.5
Dense Activation	10 Relu
Dense	1

3. Creation of the Training Set & Training Process

To capture good driving behavior,

First I try to take data by driving the car in track but due to poor driving skill in game I am unable to gain good data. My model is not performing up to mark using these data.

I first recorded two laps on track one using center lane driving.

I then recorded the vehicle recovering from the left side and right sides of the road back to center so that the vehicle would learn to came back the middle of lane.

I also drive the opposite direction of road to gain more data.

To augment the data sat, I also flipped images and angles thinking that this would help gain more data for model.

After the collection process, I had around 6539 number of data points. I then preprocessed this data by normalizing it using keras lamda function and cropping the image to remove unnecessary data.

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 5 as evidenced by more epoch does not improving the model. I used an adam optimizer so that manually training the learning rate wasn't necessary.