Behavioral Cloning Project Report

Behavrioal 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.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

The inspiration for the implemented model is derived from the NVIDIA paper about the similar problem. The model is slightly modified to cater the needs of the current customized situation. The model includes RELU layers to introduce nonlinearity, and the data is normalized in the model using a Keras lambda layer.

A summary of the network is shown in the figure below:

Layer (type)	Output Shape	Param #	Connected to
lambda_10 (Lambda)	(None, 160, 320, 3)	0	lambda_input_10[0][0]
cropping2d_10 (Cropping2D)	(None, 65, 320, 3)	0	lambda_10[0][0]
convolution2d_46 (Convolution2D)	(None, 31, 158, 24)	1824	cropping2d_10[0][0]
convolution2d_47 (Convolution2D)	(None, 14, 77, 36)	21636	convolution2d_46[0][0]
convolution2d_48 (Convolution2D)	(None, 5, 37, 48)	43248	convolution2d_47[0][0]
convolution2d_49 (Convolution2D)	(None, 3, 35, 64)	27712	convolution2d_48[0][0]
convolution2d_50 (Convolution2D)	(None, 1, 33, 64)	36928	convolution2d_49[0][0]
flatten_10 (Flatten)	(None, 2112)	0	convolution2d_50[0][0]
dropout_1 (Dropout)	(None, 2112)	0	flatten_10[0][0]
dense_21 (Dense)	(None, 100)	211300	dropout_1[0][0]
dropout_2 (Dropout)	(None, 100)	0	dense_21[0][0]
dense_22 (Dense)	(None, 50)	5050	dropout_2[0][0]
dropout_3 (Dropout)	(None, 50)	0	dense_22[0][0]
dense_23 (Dense)	(None, 10)	510	dropout_3[0][0]
dense_24 (Dense)	(None, 1)	11	dense_23[0][0]

2. Attempts to reduce overfitting in the model

The model contains dropout layers after flattening and fully connected layers in the network in order to reduce overfitting.

The model was trained and validated on the dataset I recorded to ensure that the model works in almost every situation and 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.

4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, the same has been implementing in the augmentation section of the code.

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

Model Architecture and Training Strategy

1. Solution Design Approach

The overall strategy for deriving a model architecture was to look for the authenticated, lighted and easy to implement network architecture. There were a lot of candidates that could be chosen. The notion of Transfer learning could also be made use of. But my gut feeling was whatever neural networks that were introduced in the Transfer Learning were not actually meant for this task. Also, there was no easy implementation in Keras as was the requirement of this project.

Therefore, among all the candidate networks I chose the NVIDIA architecture. I was surprised to read the paper as it was written in very simple and easy to understand language. I understand the nitty gritty of the architecture pretty well and also there was no help on how to implement in Keras, I did that with not much effort. All the time and effort was utilized in tuning some parameters and recording the right set of data.

In order to gauge how well the model was working, I split my image and steering angle data into a training and validation set. I used the 80/20 approach in order to using the 20% of training data to be used as validation data.

To combat the overfitting, I modified the model by adding the dropout layers after the flattening and fully connected layers.

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 tried to go off-track but recovered itself from leaving the track. Very rarely it also undergoes the abnormal circumstances.

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 an NVIDIA inspired convolution neural network with the following layers and layer sizes.

Layer (type)	Output Shape	Param #	Connected to
lambda_10 (Lambda)	(None, 160, 320, 3)	0	lambda_input_10[0][0]
cropping2d_10 (Cropping2D)	(None, 65, 320, 3)	0	lambda_10[0][0]
convolution2d_46 (Convolution2D)	(None, 31, 158, 24)	1824	cropping2d_10[0][0]
convolution2d_47 (Convolution2D)	(None, 14, 77, 36)	21636	convolution2d_46[0][0]
convolution2d_48 (Convolution2D)	(None, 5, 37, 48)	43248	convolution2d_47[0][0]
convolution2d_49 (Convolution2D)	(None, 3, 35, 64)	27712	convolution2d_48[0][0]
convolution2d_50 (Convolution2D)	(None, 1, 33, 64)	36928	convolution2d_49[0][0]
flatten_10 (Flatten)	(None, 2112)	0	convolution2d_50[0][0]
dropout_1 (Dropout)	(None, 2112)	0	flatten_10[0][0]
dense_21 (Dense)	(None, 100)	211300	dropout_1[0][0]
dropout_2 (Dropout)	(None, 100)	0	dense_21[0][0]
dense_22 (Dense)	(None, 50)	5050	dropout_2[0][0]
dropout_3 (Dropout)	(None, 50)	0	dense_22[0][0]
dense_23 (Dense)	(None, 10)	510	dropout_3[0][0]
dense_24 (Dense)	(None, 1)	11	dense_23[0][0]

3. Creation of the Training Set & Training Process

To capture good driving behavior, I first recorded two complete lap on track one using center lane driving. To avoid being an expert driver, the dwindling of the car was also produced Here is an example image of 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 recover in a situation where the car needs to take an abrupt change in lane either from left to center and vice versa or from right to center and vice versa.

To augment the dataset, I also flipped images and angles by adding a correction factor of about +/- 0.2 degrees for right and left camera respectively. The image is also cropped so that the network on learns what is relevant to learn. During the experimentation phase, I did more modifications like change the duration of the day, from dawn to dusk by varying the image brightness. There is very good information found from various forums by our seniors from October and November cohorts.

After the collection process, I had about ˙ 2142 (80 % trg / 20 % validate) number of data points.

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. I used an Adam optimizer so that manually training the learning rate wasn't necessary.

All the required weights are saved in the accompanying model.h5 file.