writeup_template

Behavioral Cloning

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

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 24 and 48 (model.py lines 115-117).
And then a convolution neural network with 3x3 filter sizes and depths of 64 (model.py lines 118-119)

The model includes RELU layers to introduce nonlinearity (code line 115), and the data is normalized in the model using a Keras lambda layer (code line 112).

2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce overfitting (model.py lines 116).

The model was trained and validated on different data sets to ensure that the model was not overfitting (code line 135). 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 134).

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 Nvdia's I thought this model might be appropriate because it is a real time autopilot algorithm used in Nvdia.

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 found that my first 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.

To combat the overfitting, I add more training data generated by the simulator, so that it has a similar loss on both training set and validation set.

Then I run the model for more epochs and fine tuning the model.

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, when it comes to the turning corner, to improve the driving behavior in these cases, I make more data around the corners.

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 110-126) consisted of a convolution neural network with the following layers and layer sizes.

Here is a visualization of the architecture (note: visualizing the architecture is optional according to the project rubric)

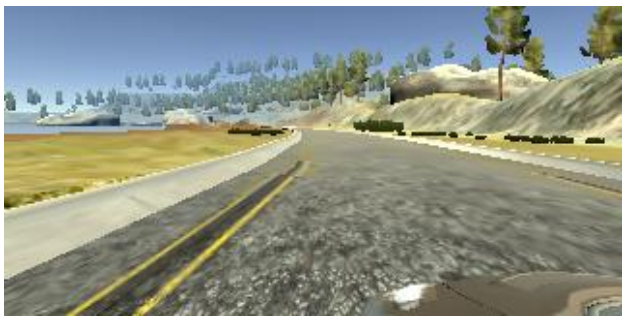
Layer (type)	Output Shape	Param #
lambda_4 (Lambda)	(None, 160, 320, 3)	0
conv2d_16 (Conv2D)	(None, 78, 158, 24)	1824
dropout_1 (Dropout)	(None, 78, 158, 24)	0
conv2d_17 (Conv2D)	(None, 37, 77, 36)	21636
conv2d_18 (Conv2D)	(None, 17, 37, 48)	43248
conv2d_19 (Conv2D)	(None, 15, 35, 64)	27712
conv2d_20 (Conv2D)	(None, 13, 33, 64)	36928
flatten_4 (Flatten)	(None, 27456)	0
dense_16 (Dense)	(None, 1024)	28115968
dense_17 (Dense)	(None, 100)	102500
dense_18 (Dense)	(None, 50)	5050
dense_19 (Dense)	(None, 10)	510
dense_20 (Dense)	(None, 1)	11
Total params:	28,355,387	
Trainable params:	28,355,387	
Non-trainable params:	0	

---

3. Creation of the Training Set & Training Process

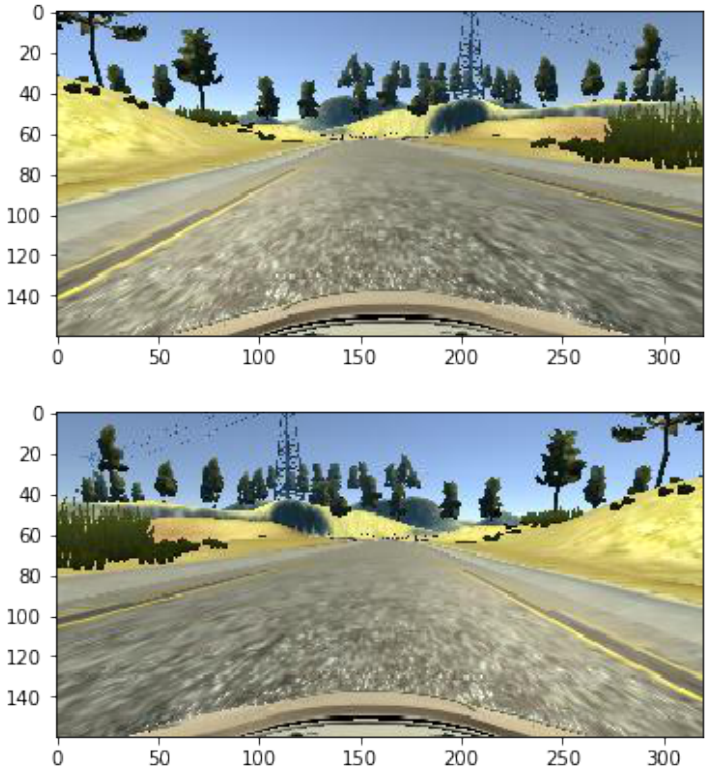
To capture good driving behavior, I first recorded two laps on track one using center lane driving. Here is an example image of center lane driving:

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 use both cameras on the side. These images show what a recovery looks like starting from corner :



Then I repeated this process on track two in order to get more data points.

To augment the data sat, I also flipped images and angles thinking that this would help the car to learn more driving direction. For example, here is an image that has then been flipped:



After the collection process, I had 48135 number of data points. I then preprocessed this data by flipping it, it change the data to the double size.

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 10 as evidenced by earlystopping, which the vac_loss didn't decrese for two epochs. I used an adam optimizer so that manually training the learning rate wasn't necessary.