Project Writeup

Project title: Behavior cloning

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
- video.mp4 containing two laps of autonomous driving at 30 speed
- writeup_report.pdf summarizing the results

Additional details on the project can be found in:

GitHub repository: Juptyer notebook
github.com/sibyjackgrove/Car_driving_Behaviour_cloning/blob/master/clone_driving_CNN-rev1.ipynb

YouTube: 3rd Person view

https://www.youtube.com/watch?v=kfIqHTmHmes

Solution approach: End to end deep learning using Convolutional neural network

The model.py file contains the code for training and saving the convolution neural network. The drive.py file was modified by me so that it will work with TensorFlow's built in Keras module instead of the standard Keras module.

Model Architecture and Training Strategy

1) The model (including input pipeline) was built using TensorFlow's built in Keras module.
   Input pipeline:
   The model takes as input a 160×320×3 image tensor (model.py line 83) which is first cropped along vertical axis to obtain a 90×320×3 tensor (model.py line 84) and then resized into a 66×66×3 tensor. (model.py line 85) The Nearest Neighbor resize method available in TensorFlow is used. The values in the image tensor are then normalized Keras lambda layer so that they lie between -0.5 to 0.5.

   Deep Neural Network:
   The model consists of four convolutional layers (model.py lines 88 to 98), followed by two feed-forward (fully connected) layers (model.py lines 102-106), and finally the output layer. The convolutional layers and feed forward layers use ReLU activation to introduce non-linearity while the output layer uses linear activation function.
   The weights (kernel) in all the layers are initialized using the "glorot_uniform" initializer function which is default initializer. The biases in all the layers were initialized with zeroes. Total model had a total of 4299 trainable parameters (weights + biases).

The below table shows the details of the complete model.

Layer (type)	Output Shape	Param #
InputLayer (InputLayer)	(None, 160, 320, 3)	0
CropImage (Cropping2D)	(None, 90, 320, 3)	0
ResizeImage (Lambda)	(None, 66, 66, 3)	0
Normalizing (Lambda)	(None, 66, 66, 3)	0
ConvLayer1 (Conv2D)	(None, 31, 31, 4)	304
ConvLayer2 (Conv2D)	(None, 14, 14, 6)	606
ConvLayer3 (Conv2D)	(None, 5, 5, 8)	1208
DropOut1 (Dropout)	(None, 5, 5, 8)	0
ConvLayer4 (Conv2D)	(None, 1, 1, 10)	2010
Flatten (Flatten)	(None, 10)	0
FeedForward1 (Dense)	(None, 10)	110
ReLU1 (Activation)	(None, 10)	0
FeedForward2 (Dense)	(None, 5)	55
ReLU2 (Activation)	(None, 5)	0
OutputLayer (Dense)	(None, 1)	6

Training:
The model was trained using Adam optimizer algorithm (model.py line 109) with mean square error as loss function. Error refers to the difference in actual versus predicted steering angle. The training parameters are:
Mini-batch size: 32
Epochs: 3
Training samples: 24407
Validations samples: 6102

2) Attempts to reduce overfitting in the model.
The model contains dropout layer after convolutional layer in order to reduce overfitting (model.py line 96).
The original data set was split between 80% (for training) and 20% for (validation) (code line 114). 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 109).

4) Appropriate training data
   I drove the car for several laps in the simulator, trained the model, and tested it in autonomous mode. I only used the center camera image for the training input. I noted parts of the track the model was not performing well, and generated training data for that lap. I could incrementally improve the performance of the model in this way. Examples of training data are given below (center camera image and corresponding steering angle):

Steering angle: -0.03529 Steering angle: -0.08235 Steering angle: 0.005882

Conclusion:

The end to end deep learning model could successfully replicate driving behavior. It could keep the car inside the lane markings, negotiate the turns, and complete the track indefinitely.