Behavioral Cloning

Files Submitted

File/Directory name	Description
cloning.py	This file contains the code for creating the Behavioral Cloning model
model.h5	This file is the model created by running cloning.py
drive_data	This directory contains driving_data.csv and image files. These were created in Simulation mode
run1	This directory contains images of the vehicle driving in Autonomous mode
run1.mp4	This file is the final video of the vehicle driving in Autonomous mode

Data Collection

This project taught me the importance of Quality vs Quantity of data.

First, I followed suggestions in the lesson and captured one lap each of driving the Track 1 in forward and reverse directions. Then, I tested my model and the vehicle went into the water.

Then, I collected some data of the vehicle recovering from the edges. This improved the final output. However, I still had a problem where the vehicle went off the road and into the dirt section at one of the left turns.

So then, I thought that I need more data, and I collected 2 laps of forward and reverse driving, and 1 lap of recovery data in both forward and reverse direction. In all this data, I was driving very cautiously, and maintaining center lane driving. The result was that the vehicle was driving close to the right edge while making left turns, and close to the left edge when making right turns. At one point, it went very close to the right edge and hit a pillar near the bridge and got stuck. So, I collected even more data, and ended up getting Memory Error.

This is when I realized that quality matters more than quantity. So, I captured one lap of driving in the center lane, a lot of recovery data, and what I think is the most important was recovery data where I drove the vehicle from the outside edge to the inside edge, like race car drivers trying to drive across the apex of the turn. My final output shows this behavior, and it can be made smoother if I collect some more data like this. In the real world, driving an autonomous vehicle only in the center lane will not always be smooth and will result in more roll, unless the speed is reduced before and while turning. Driving across the apex will allow for maintaining the speed and reducing roll, therefore resulting in a smoother ride.

Training and Validation data sets

From the collected driving data, I decided to find out how much data belongs to left, right and straight driving. For this purpose, I decided to check the 4th column to analyze at the steering angle and decide if it is a left turn when steering angle < -0.12, a right turn when steering angle > 0.12, and straight driving otherwise.

The data collected had the following statistics:

Left turns = 1630

Right turns = 404

Straights = 5866

As the track has more left turns than right turns, the above numbers are to be expected, i.e. significant number of data points for left turns as compared to right turns.

Therefore, I decided to augment this data by flipping the images and corresponding angles. This would get me more data without having to drive around in Simulation mode. I then used this augmented data set to Train/Test and Split the data.

I decided to use 90% of the data for the Training set and 10% of the data for the Validation set.

Here are the statistics after training the data:

Left turns = 5412

Right turns = 5435

Straights = 31813

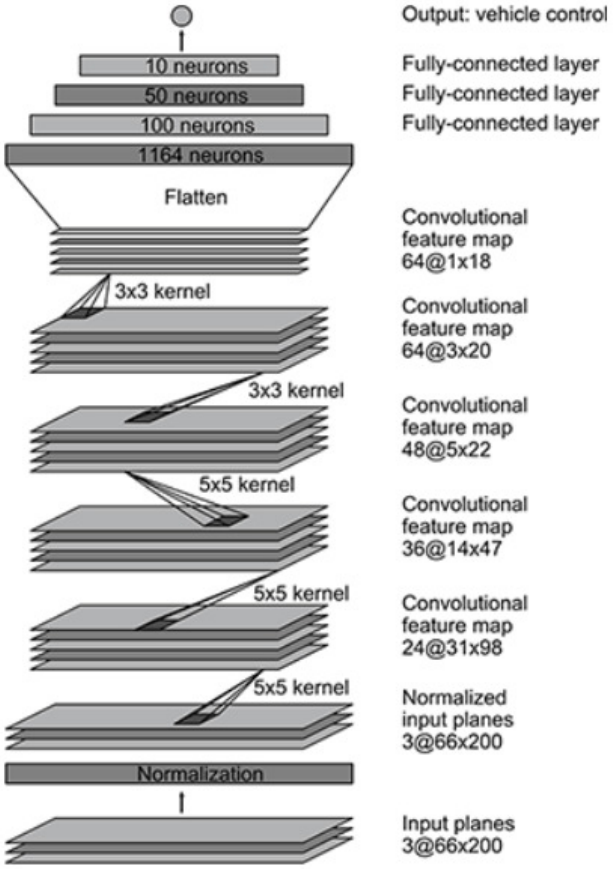
Now I have 3 times more data than before, and the numbers for left and right turns are comparable.

Model Architecture and Training Strategy

The model was a Convolutional Neural Network, using the NVIDIA model as a starting point (http://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf) with some modifications.

The input shape used is 160x320x3.

I added a Dropout layer with dropout rate as 0.5 after the Flatten layer to reduce overfitting.



In the model fit method, I used shuffle=True to allow shuffling by Keras.

I added a Dropout layer with dropout rate as 0.5 after the Flatten layer to reduce overfitting. The model was tested by running it through the simulator to make sure that the vehicle stays on the track.

Here is the model summary:

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 160, 320, 3)	0
cropping2d_1 (Cropping2D)	(None, 65, 320, 3)	0
conv2d_1 (Conv2D)	(None, 31, 158, 24)	1824
conv2d_2 (Conv2D)	(None, 14, 77, 36)	21636

conv2d_3 (Conv2D)	(None, 5, 37, 48)	43248
conv2d_4 (Conv2D)	(None, 3, 35, 64)	27712
conv2d_5 (Conv2D)	(None, 1, 33, 64)	36928
flatten_1 (Flatten)	(None, 2112)	0
dropout_1 (Dropout)	(None, 2112)	0
dense_1 (Dense)	(None, 100)	211300
dense_2 (Dense)	(None, 50)	5050
dense_3 (Dense)	(None, 10)	510
dense_4 (Dense)	(None, 1)	11

Total params: 348,219

Trainable params: 348,219

Non-trainable params: 0

Train on 42660 samples, validate on 4740 samples

Model parameter tuning

I used the Adam optimizer, so the learning rate was not tuned manually.

Model performance

I started with 5 epochs and found that loss was reducing but the vehicle movements were not very smooth. Therefore, I decided to increase the number of epochs to 10. There was not much improvement between epochs 9 and 10, therefore I settled on 9 epochs. Each epoch too approximately 66 seconds to execute.

42660/42660 [==============================] - 62s 1ms/step - loss: 0.0605 - acc: 0.2368 - val_loss: 0.0543 - val_acc: 0.2291

Epoch 2/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0477 - acc: 0.2377 - val_loss: 0.0453 - val_acc: 0.2306

Epoch 3/9

42660/42660 [==============================] - 65s 2ms/step - loss: 0.0390 - acc: 0.2399 - val_loss: 0.0357 - val_acc: 0.2329

Epoch 4/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0322 - acc: 0.2411 - val_loss: 0.0314 - val_acc: 0.2335

Epoch 5/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0273 - acc: 0.2420 - val_loss: 0.0276 - val_acc: 0.2344

Epoch 6/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0239 - acc: 0.2423 - val_loss: 0.0235 - val_acc: 0.2348

Epoch 7/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0216 - acc: 0.2423 - val_loss: 0.0221 - val_acc: 0.2350

Epoch 8/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0202 - acc: 0.2425 - val_loss: 0.0221 - val_acc: 0.2350

Epoch 9/9

42660/42660 [==============================] - 66s 2ms/step - loss: 0.0184 - acc: 0.2426 - val_loss: 0.0199 - val_acc: 0.2350

Further improvements

This is a very interesting project. The driving behavior can be improved with better quality of data, especially recovery scenarios. Data can be augmented with randomly changing the brightness of data collected to have a data set with varying levels of brightness. Also, data from left and right cameras could be used to help the model train better.