Behavioral Cloning

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

My project includes the following files:

- model.py containing the script to create and train the model
- model.ipynb containing the jupyter notebook version of model.py 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.ipynb and writeup_report.pdf summarizing the results

Testing

- 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
```

- Saving the autonomous run can be done by executing:

```
python drive.py model.h5 <filename>
```

Model Architecture and Training Strategy

I used the same network architecture proposed in NVIDIA's "End to End Learning for Self-Driving Cars" paper.

Model Architecture

The model is exactly the same as the proposed NVIDIA model, but I added a dropout layer before the second last layer, to prevent overfitting and for the model to generalize better.

Layer 1: Normalization Layer.
Kernel: nil
Stride: nil
The output shape: 66x200x3.

Layer 2: Convolutional.
Kernel: 5x5x3x24
Stride: 2x2
The output shape: 31x98x24.

Layer 3: Convolutional.
Kernel: 5x5x24x36
Stride: 2x2
The output shape: 14x47x36.

Layer 4: Convolutional.
Kernel: 5x5x36x54
Stride: 2x2
The output shape: 5x22x54.

Layer 5: Convolutional.
Kernel: 3x3x54x64
Stride: 1x1
The output shape: 3x20x64.

Layer 6: Convolutional.
Kernel: 3x3x64x64
Stride: 1x1
The output shape: 1x18x64.

Flatten.
Flattens the output shape of the final pooling layer such that it's 1D instead of 3D.
Output = 1152

Layer 1: Fully Connected.
Layer Operation = 1x1152 x 1152x1164
Layer Output = 1x1164

Activation.
ReLU activation.

Layer 2: Fully Connected.
Layer Operation = 1x1164 x 1164x100
Layer Output = 1x100

Activation.
ReLU activation.

Layer 3: Fully Connected.
Layer Operation = 1x100 x 100x50
Layer Output = 1x50

Activation.
ReLU activation.

Layer 4: Dropout Layer

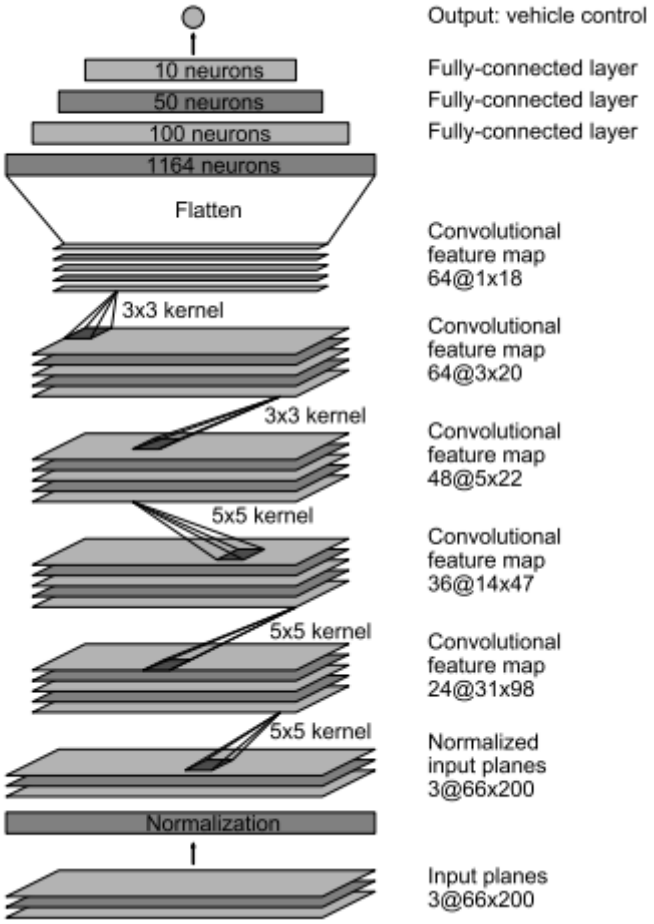
Layer 5: Fully Connected.
Layer Operation = 1x50 x 50x10
Layer Output = 1x10

Activation.
ReLU activation.

Layer 6: Fully Connected.
Layer Operation = 1x10 x 10x1
Layer Output = 1

Output

1 outputs



Source: End to End Learning for Self-Driving Cars (NVIDIA Corporation)

Training Strategy

Creation of the Training Set

1. Initially, I just used the sample training data provided.

- First, I just used the image data from center camera, and was met with very limited success. The car would barely, move before crashing onto the side.
- Then, I included the image data from both left and right cameras. I included a correction factor to the steering data, when left and right image data was used. This correction factor had to be fine-tuned. The best result I got was with a correction factor of 0.05.
- I was able to make the car go past the first curve, but It ended up crashing in the next one. I tried a lot of tuning the network parameters, but was still met with limited success.

2. Then I collected my own dataset.

- It took sometime to collect the dataset, since I had to get used to controlling the car in the simulator.
- Initially, I only collected the data driving only on the center of the road as much as possible. (3-Laps)
- After, following the same stratergy of including the left and right cameras and taking into the steering correction. The car was able to drive close to half-way before crashing at the turn after bridge. This was mainly, because I have not yet collected the data for the network to learn, when they are close to edge.
- As a final stratergy, I collected the data for recovering driving from the sides and turns. I also collected a single lap of center driving data from track two to make a more generalized model. The final collected dataset is given in the IMG folder and video output 1 is shown below.
- After training the model, My car was able to drive smoothly across the lap, and even though it side tracked at times, it was able to recover back to the center.
- The trained weights have been included in set-1. I have included the output in video 2.

3. Lastly, I combined my collected dataset and the sample provided dataset.

- This was done in order to see, if the driving could be made more efficient.
- The car was able to drive without crashing, I have included the output in video-3. The car was much more at the center.
- The trained weights have been included in set-2.

Training Process

Type of optimizer:

Adaptive Moment Estimation (Adam) Optimizer was to used to optimize the model. The learning rate was set to 0.0001. The other parameters for adam optimizer was set as the deafult parameters. Adam optimizer was used because it works well in practice and compares favorably to other adaptive learning-method algorithms. In addition to storing an exponentially decaying average of past squared gradient like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients similar to momentum.

Cost Function:

Mean Squared Error.

Batch size:

I used the batch size of 256. I tried 128 and 64 batch sized too, but 256 was giving me a better result.

Epochs:

The model was trained for 5 epochs. I found that around 4 or 5 epochs, the validation loss was staying almost constant. Hence, I selected the model to train for 5 epochs.

In [4]:
```
from moviepy.editor import VideoFileClip
from IPython.display import HTML
```

Collected Dataset

In [11]:
```
collected_dataset = 'examples/IMG.mp4'
```

In [12]:
```
HTML("""
<video width="320" height="200" controls>
  <source src="{0}">
</video>
""".format(collected_dataset))
```

Out[12]:

Autonomous Driving after training only on the collected dataset

In [13]:
```
car_output1 = 'examples/run1.mp4'
```

In [14]:
```
HTML("""
<video width="320" height="200" controls>
  <source src="{0}">
</video>
""".format(car_output1))
```

Out[14]:

Autonomous Driving after training on both collected dataset and sample training dataset

In [15]:
```
car_output2 = 'examples/run2.mp4'
```

In [16]:
```
HTML("""
<video width="320" height="200" controls>
  <source src="{0}">
</video>
""".format(car_output2))
```

Out[16]: