Behavioral Cloning

This project was part of Udacity's Self Driving Car Nano Degree Program.The goal of project was to train the car to drive itself in the video game.The car was trained to drive itself using Deep neural network namely convolutional layers with automated feature engineering.

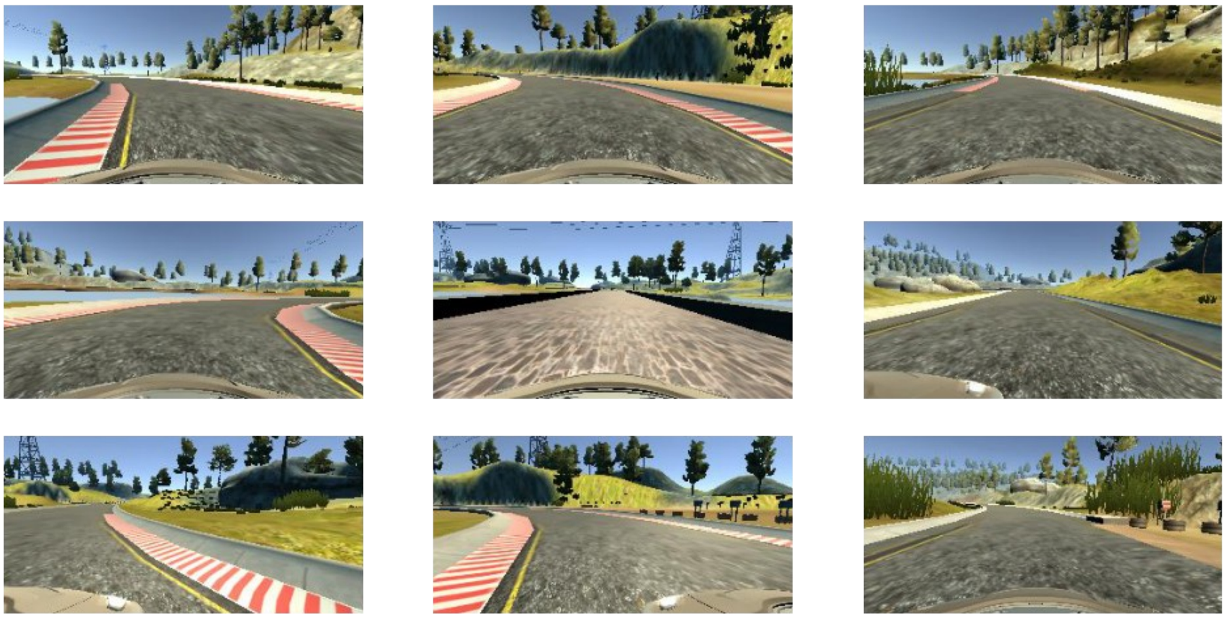
Files Included

- model.py - The script used to train the model.
- drive.py - The script to drive the car.
- model.h5,model.json - model weights and config
- run1,run2- video for simulation running on track 1 and track 2

Exploring the datasets

I used the dataset provided by the Udacity and I haven't recorded images myself.

The dataset contains the JPG images of dimensions 160*320*3.



Model Architecture Design

Model is designed based on NVIDIA architecture.

I have cropped images in model itself because we don't want to train it on skies and car hood, secondly we will not require to do any modification in the drive.py while running car in simulation.

After the above step I have normalized the input images in order to avoid saturation and make gradients work better.

I also applied droput after the convolution layer to handle overfitting and activation function was applied in each layer except for final layer to introduce nonlinearity.

The model was tested by running it through the simulator, ensuring that the vehicle could stay on the track.

In the end model looks like this

- Image cropping
- Image normalization
- Convolution: 5x5, filter: 24, strides: 2x2, activation: ELU
- Convolution: 5x5, filter: 36, strides: 2x2, activation: ELU
- Convolution: 5x5, filter: 48, strides: 2x2, activation: ELU
- Convolution: 3x3, filter: 64, strides: 1x1, activation: ELU
- Convolution: 3x3, filter: 64, strides: 1x1, activation: ELU
- Drop out (0.2)
- Fully connected: neurons: 100, activation: ELU
- Fully connected: neurons: 50, activation: ELU
- Fully connected: neurons: 10, activation: ELU
- Fully connected: neurons: 1 (output)

Below is architecture design of the model

```
____________________________________________________________________________________
Layer (type)                 Output Shape          Param #     Connected to
____________________________________________________________________________________
cropping2d_1 (Cropping2D)        (None, 65, 320, 3)    0           cropping2d_input_1[0][0]
____________________________________________________________________________________
lambda_1 (Lambda)                (None, 65, 320, 3)    0           cropping2d_1[0][0]
____________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 31, 158, 24)   1824        lambda_1[0][0]
____________________________________________________________________________________
```

```
convolution2d_2 (Convolution2D)  (None, 14, 77, 36)    21636       convolution2d_1[0][0]

convolution2d_3 (Convolution2D)  (None, 5, 37, 48)     43248       convolution2d_2[0][0]

convolution2d_4 (Convolution2D)  (None, 3, 35, 64)     27712       convolution2d_3[0][0]

convolution2d_5 (Convolution2D)  (None, 1, 33, 64)     36928       convolution2d_4[0][0]

dropout_1 (Dropout)              (None, 1, 33, 64)     0           convolution2d_5[0][0]

flatten_1 (Flatten)              (None, 2112)          0           dropout_1[0][0]

dense_1 (Dense)                  (None, 100)           211300      flatten_1[0][0]

dense_2 (Dense)                  (None, 50)            5050        dense_1[0][0]

dense_3 (Dense)                  (None, 10)            510         dense_2[0][0]

dense_4 (Dense)                  (None, 1)             11          dense_3[0][0]
```

Training Parameter

1. Optimizer: Adam Optimizer
2. No. of epochs: 5
3. Batch_size: 64
4. Kera's fit_generator method was used to train images generated by the generator
5. Success of model was evaluated by how well is drives on the road and not by loss function.

Data Augmentation Strategy

- For each data point we randomly choose camera position and adjust the steering angle by 0.25
- we also randomly flip the images and change the sign of steering angle to generate more data.

The model works perfectly fine for Lake -Side track but failed for Jungle track after few minutes of driving.

For the jungle track we need to extract the images from training mode and train the model using them as the turns are very tricky to be learnerd.