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

My project includes the following files:

- **P3BehavioralCloning .ipynb** 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
- **Behavioral Cloning Project.pdf** summarizing the results
- **video_output_fantastic.mp4**, **video_output_fastest.mp4** are the output files

The **P3BehavioralCloning .ipynb** file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

My model (code cell #5) consists of a convolution neural network with 5x5 & 3x3 filter sizes and depths between 24 and 64. And the model summary can be seen below code cell #5

The model includes ELU layers to introduce nonlinearity after every convolution layer, and the data is normalized in the model using a Keras lambda layer (code line 11).

The model contains dropout layers after every 'Dense' layer (except the last one) with keep probability of 50% in order to reduce overfitting.

I used train_test_split to use 20% of center camera images to validate the trained model. The model was tested by running it through the simulator and ensuring that the vehicle could stay on the track.

The model used an adam optimizer, so the learning rate was not tuned manually

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, I added offset to left and right camera images in generator (code cell #3) to make sure that model learns recovery from sliding off the road.

I have used convolution network from NVIDIA end-to-end learning paper. As this was a proven approach to drive the vehicle, I finalized this. Though I tried different architectures, this one gave me a satisfied test results.

In the initial iterations, I used a different approach to augment the data, preprocess the data, and I get many data samples(images) after augmentation which made the network to overfit because of sheer amount of data and low depth of network. I finally settled for below two augmentations,

- Flipping images: This is to make sure that vehicle learns both left and right turns, for this, in my generator, I randomly flipped some images, and its angles.
- Adding offset to left and right cameras: This is make sure vehicle learns how to get back to center when it's about to slide off the road.

Using just Udacity provided training data set along with above augmentations, car was able to drive across the first track fine.

Overall this is the hardest project of all 5 projects in Term 1. Because it requires a lot of guess and check work as well as good training data. I learnt from this project is that data is the winner over network because if you through in garbage data at network, network will output garbage data as output.

Next step is to augment the data using functions shown in (code cell #2) and try to make the model work for Mountain Track.