Autonomous Driving Project # 4 – Behavior Cloning

Himanshu Kumar

Project Definition:

This project has following requirements:

- Code should be functional & generate training data
- Model architecture & training strategy
- Architecture & training documentation
- Simulation

Below section will explain each point in detail.

Generating Training Data:

For this project, capturing correct training data was critical for training the network. I captured training data for these scenarios:

- Driving in the middle for 2 laps on the track. (Folder name = "trainingData2Lap")
- Driving in the middle for tough track. (Folder name = "toughTrackTrainingData")
- Capturing training data for recovering car from edge of the track to middle of the track. This data proved to be critical in training. I included this data twice for increasing the number of samples for recovery path in training. (Folder name = "trainingDataRecovery")
- Training data driving clockwise on track. (Folder name = "trainingDataClockWise")
- And I also used sample training data provided. (Folder name = "behaviorCloningSampleData")

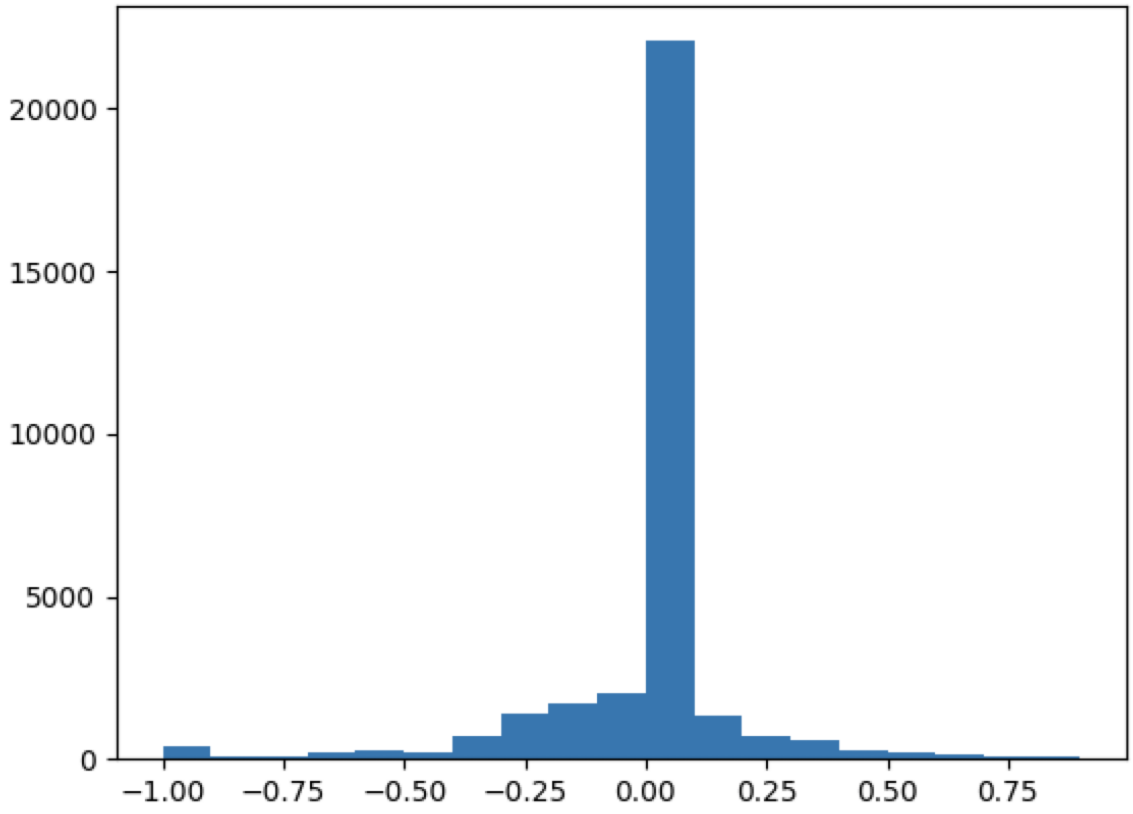
I only used center camera image for training.

Exploring Data Set:

I did some exploration of the data set to get idea about some of the stats, which are below (code is in KerasLeNet.py):

Total number of training images (only center image): 32697

Distribution of steering angles in training data:



Distribution of different types of training data (only center image):

Tough Track Training Images: 2588
Simple Track Training Images: 16194
Recovery from edge Training Image: 8428
Simple Track Clock wise Training Images: 5489

Model Architecture and Training Strategy:

I tried LeNet architecture for behavior cloning but it did not perform well on sharp turns, for example after crossing bridge, it failed to take sharp turn to avoid dirt on the side of the track.

Then I tried autonomous driving neural network architecture provided by NVIDIA. I was surprised by the performance of this network. Even with only a small portion of training data, it outperformed LeNet architecture which was trained on full training data. Then I decided to pursue this model. I trained the model on my training data and it performed very well. It was very well in predicting steering angles for sharp turns.

Some of the feature of this model:

1. Training was very fast (in comparison to LeNet)
2. It seems, this network can do better than even if we have less training data
3. Number of epochs required was only 2.

Here is the model:

Layer	Description
Input	RGB image (160x320x3)
Convolution 1	filter size = 5, Stride = 2, padding = Valid, and number of filters is 24, input channel = 3. This gives output of 78x158x24
Activation	RELU
Convolution 2	filter size = 5, Stride = 2, padding = Valid and number of filters is 36, input channel = 24. This gives output of 37x77x36
Activation	RELU

Convolution 3	filter size = 5, Stride = 2, padding = Valid and number of filters is 48, input channel = 36. This gives output of 17x37x48
Activation	RELU
Convolution 4	filter size = 3, Stride = 1, padding = Valid and number of filters is 64, input channel = 48. This gives output of 15x35x64
Activation	RELU
Convolution 5	filter size = 3, Stride = 1, padding = Valid and number of filters is 64, input channel = 64. This gives output of 13x33x64
Activation	RELU
Flatten	27456
Fully Connected	Input is 27456 and output is 100 (in logits)
Activation	RELU
Fully Connected	Input is 100 and output is 50 (in logits)
Activation	RELU
Fully Connected	Input is 50 and output is 10 (in logits)
Activation	RELU
Fully Connected	Input is 10 and output is 1 (in logits)

Things which I tried but did not worked out so I end up removing it from model:

- Drop out layer
- Flipping the training image

The hyperparameters for this model:

- Epochs = 2
- Training/Validation Split: 80:20
- Learning rate: 0.001 (Adam optimizer default leaning rate)
- Batch size: 32 (Keras model.fit default value)

Pre-processing on image:

- Cropping image. For cropping, I choose 70 from top and 25 from bottom. No cropping of image from left and right.
- Normalization of image.

Simulation:

After training, I saved the model as “model.h5”. The output video of the autonomous driving is saved as “finalRun.mp4”. Here is the video:

Future Improvement:

I think, there are few things which we can try for improving the performance of this model:

- Converting image to different color space. NVDIA has used YUV plane instead of RGB which is used my model.
- I used default image size of 160x320x3 instead of image size use by NVDIA (66x200x3).
- Adding images from multiple camera instead of images from center camera only as used by my model.
- Other network like AlexNet might work better as suggested by this paper (https://arxiv.org/pdf/1803.09386.pdf).