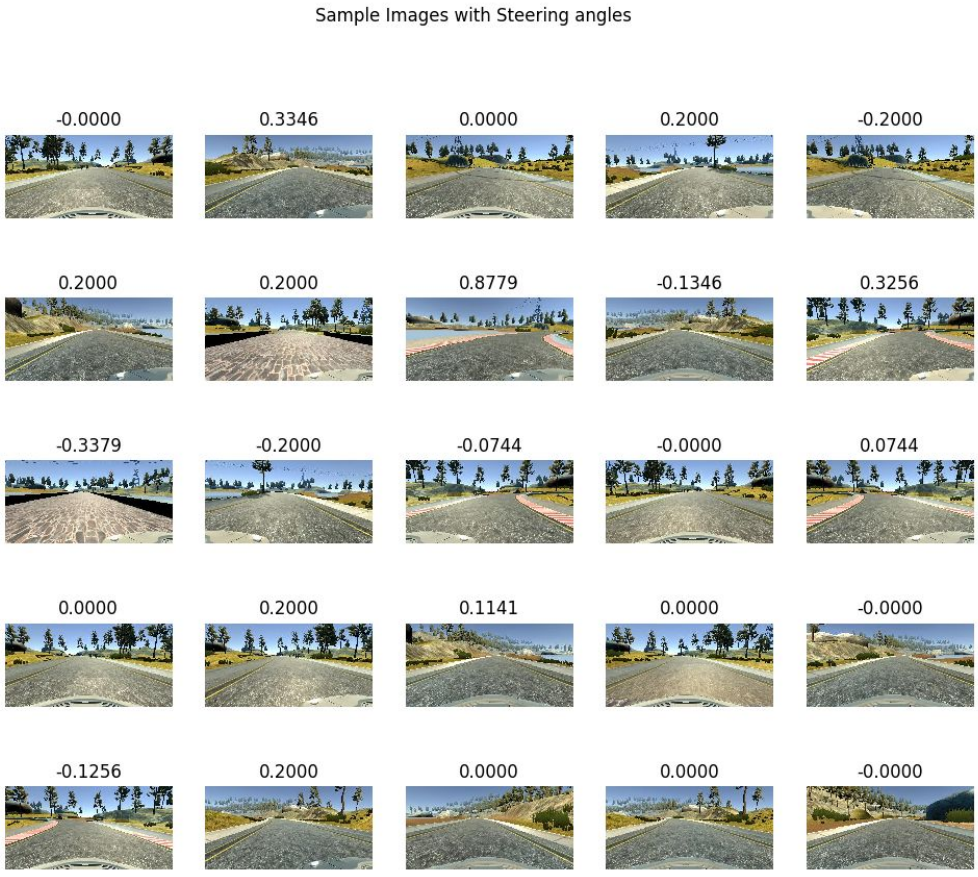
CODE -

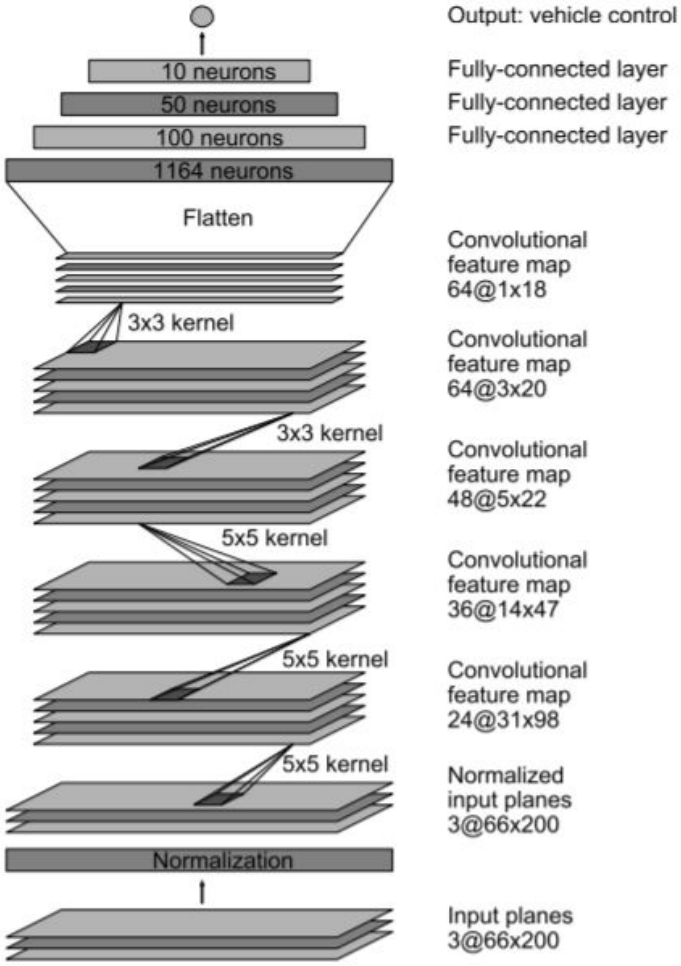
The code has been distributed into two files -

- Model.py - Contains the model and training routine
- Dataloader.py - Contains code to return a Python generator to load training data into memory. Running the file gives a sense of what the data looks like
  Example output -



MODEL AND TRAINING -

The model used is the NVIDIA Net, with a slight dropout added just after the Flatten layer. This helped in reducing the validation loss a bit, and was also visible as bit more stable in terms of driving



The net used by the folks as Nvidia (Link to paper)

A summary of the model is as follows -

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 160, 320, 3)	0
cropping2d_1 (Cropping2D)	(None, 80, 320, 3)	0
conv2d_1 (Conv2D)	(None, 38, 158, 24)	1824
activation_1 (Activation)	(None, 38, 158, 24)	0
conv2d_2 (Conv2D)	(None, 17, 77, 36)	21636
activation_2 (Activation)	(None, 17, 77, 36)	0
conv2d_3 (Conv2D)	(None, 7, 37, 48)	43248
activation_3 (Activation)	(None, 7, 37, 48)	0
conv2d_4 (Conv2D)	(None, 5, 35, 64)	27712
activation_4 (Activation)	(None, 5, 35, 64)	0
conv2d_5 (Conv2D)	(None, 3, 33, 64)	36928
activation_5 (Activation)	(None, 3, 33, 64)	0
flatten_1 (Flatten)	(None, 6336)	0
dropout_1 (Dropout)	(None, 6336)	0
dense_1 (Dense)	(None, 100)	633700
activation_6 (Activation)	(None, 100)	0
dense_2 (Dense)	(None, 50)	5050
activation_7 (Activation)	(None, 50)	0
dense_3 (Dense)	(None, 10)	510
activation_8 (Activation)	(None, 10)	0
dense_4 (Dense)	(None, 1)	11

Total params: 770,619
Trainable params: 770,619
Non-trainable params: 0

The Adam optimizer was chosen to have the benefit of an adaptive learning rate and not tune it manually. 4 epochs seemed to be enough as the validation loss stopped decreasing.

DATA -

The data used was a sum up of the images provided by udacity and some more collected from the simulator. The data was collected in three runs.

- Center driving
- Center driving in the opposite direction
- Recording only when recovering from the corners

20 percent of the data is used for validation
Data was normalised using the Keras Lambda layer.
Images were cropped from the top and the bottom such the only the lanes remain, removing parts of the image that add more distraction than value.
The left and the right camera images were also used with adjustments in the steering angle with a correction factor of 0.2 as described in the lectures.
Additionally, each image was also flipped and its steering angle negated, doubling the whole dataset.

CONCLUSION AND IMPROVEMENTS

Bigger Net -

A bigger net may do a better job but I avoided it as it would require a huge dataset to train on. I believe Imagenet weights of something like a ZFNet would not work quite good due to the domain shift between simulator images and real world Imagenet images.

Data Bias -

The data is more biased towards a steering angle of 0.0 and thus as a basic step we could remove 20-50 percent of the dataset which contain a steering angle of 0.0.
A much better approach would be to blur out the steering angles such that the data points before and after a non-zero steering angle get some small values. This would also emulate the real world as it would resemble soft steers.