Solution Design Approach:

I started with sample dataset and a model based on nvidia self-driving model: (https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf). I then followed an iterative approach of capturing new data, training (reducing validation error) and trying the model on simulator. In case of high training loss, I would tweak the model by adding more convolutions and in case of high validation loss, I would add dropout layers or add more data. In case the model didn't work well in simulation, I captured more data or discarded old data as explained below.

Dataset creation and training process

I initially used data from Default simulator to capture driving at center of the road using keyboard and didn't get good results on autonomous simulator mode with it.

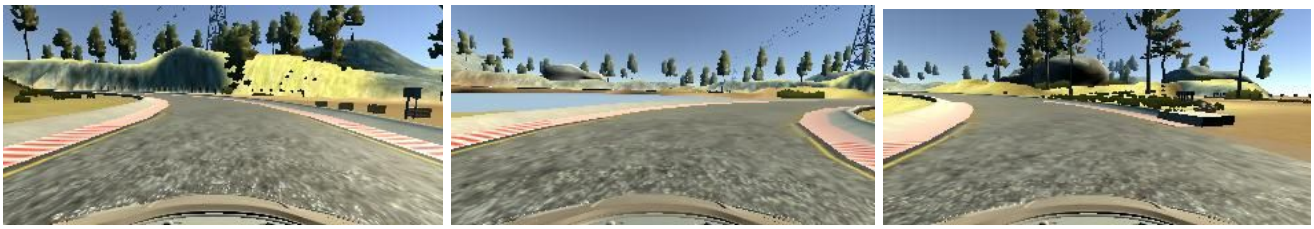
I realized since we only had keyboard input in default simulator, my left and right movements were not smooth and this might not give good results in training.
Samples showing jerky movement:
Continuous steering angles:
Snapshot1: -0.3055651, -0.5000424, -**0.09990263**, -0.4005447, -0.3123057
Snapshot2: -0.3480477,**-0.04513478, 0**, -0.1001909, -0.455409, -0.1588955

I then proceeded to capture new training data using beta simulator and controlling steering using mouse taking care to make smooth turns and avoid jerky steering. After a lot of trial and error process, I realized that best way to capture data is in short bursts which capture good data, rather in a long continuous sequence which might have bad data as well (if I steer towards the side, for example). As suggested, I captured specifically three patterns: middle of road driving, turns and recovery. Steering values also looked much better this time:
-0.06603774, -0.06603774,-0.08490566,-0.1037736,-0.1037736,-0.1037736, -0.1132075, -0.1320755, -0.1415094, -0.1415094, -0.1415094

In totality I had about ~16k steering log lines (~8k from sample, ~8k from captured using beta simulator). Some samples from training data:

Middle of road driving:

Turns:



Recovery:

Data Augmentation:

I used left and right images from sample dataset as well with correction of 0.2, and all images were also reversed (with steering negated) to increase training and validation data size. Relevant code section:

```
df = pd.read_csv('../tr/driving_log.csv')
dfc = df[['center', 'steering']].rename(columns={'center':'image'})
dfl = df[['left', 'steering']].rename(columns={'left':'image'})
dfl['steering'] = dfl['steering'] + 0.2
```

```
dfr = df[['right', 'steering']].rename(columns={'right':'image'})
dfr['steering'] = dfr['steering'] - 0.2
dfn = pd.concat([dfc, dfl, dfr])
df = dfn.dropna() # beta simulator doesn't give left/right images
df['flip'] = 0
df_flipped = df.copy()
df_flipped['flip'] = 1

df = pd.concat([df, df_flipped])
```

After augmentation and considering left and right camera images as well, I had roughly ~65k images.

Training process:
I trained on aws g2.2xlarge machine as local machine took too long. Normalization and cropping were built into the model network to leverage it in simulation as well. Relevant code section:

```
model.add(Lambda(lambda x : x/255.0 - 0.5, input_shape=(160, 320, 3), output_shape=(160, 320, 3)))
model.add(Cropping2D(cropping=((50, 20), (0,0))))
```

For training I split the data into training and validation in 80:20 ratio to figure out under/over fit. Adam optimizer was used. Batch size, number of epochs and model layers were tuned to optimize training and validation loss. I also tried optimizing both mean squared error and mean absolute error, **mean squared error** giving better results. I found **batch size of 256 and 30 epochs** to give best validation loss without overfitting too much.

Final Architecture:

Layer (type)	Output Shape	Param #	Connected to
==========	==========	==========	==========

```
lambda_1 (Lambda)               (None, 160, 320, 3)   0           lambda_input_1[0][0]
____________________________________________________________________________________________
cropping2d_1 (Cropping2D)       (None, 90, 320, 3)    0           lambda_1[0][0]
____________________________________________________________________________________________
convolution2d_1 (Convolution2D)  (None, 43, 158, 24)   1824        cropping2d_1[0][0]
____________________________________________________________________________________________
activation_1 (Activation)       (None, 43, 158, 24)   0           convolution2d_1[0][0]
____________________________________________________________________________________________
convolution2d_2 (Convolution2D)  (None, 20, 77, 36)    21636       activation_1[0][0]
____________________________________________________________________________________________
activation_2 (Activation)       (None, 20, 77, 36)    0           convolution2d_2[0][0]
____________________________________________________________________________________________
convolution2d_3 (Convolution2D)  (None, 8, 37, 48)     43248       activation_2[0][0]
____________________________________________________________________________________________
activation_3 (Activation)       (None, 8, 37, 48)     0           convolution2d_3[0][0]
____________________________________________________________________________________________
convolution2d_4 (Convolution2D)  (None, 6, 35, 64)     27712       activation_3[0][0]
____________________________________________________________________________________________
activation_4 (Activation)       (None, 6, 35, 64)     0           convolution2d_4[0][0]
____________________________________________________________________________________________
convolution2d_5 (Convolution2D)  (None, 4, 33, 128)    73856       activation_4[0][0]
____________________________________________________________________________________________
activation_5 (Activation)       (None, 4, 33, 128)    0           convolution2d_5[0][0]
____________________________________________________________________________________________
flatten_1 (Flatten)             (None, 16896)         0           activation_5[0][0]
____________________________________________________________________________________________
dropout_1 (Dropout)             (None, 16896)         0           flatten_1[0][0]
____________________________________________________________________________________________
dense_1 (Dense)                 (None, 128)           2162816     dropout_1[0][0]
____________________________________________________________________________________________
dense_2 (Dense)                 (None, 64)            8256        dense_1[0][0]
____________________________________________________________________________________________
dense_3 (Dense)                 (None, 16)            1040        dense_2[0][0]
____________________________________________________________________________________________
dense_4 (Dense)                 (None, 1)             17          dense_3[0][0]
============================================================================================
Total params: 2,340,405
Trainable params: 2,340,405
```

Non-trainable params: 0