In [1]:

```
# Credits: https://github.com/SullyChen/Autopilot-TensorFlow
# Research paper: End to End Learning for Self-Driving Cars by Nvidia. [https://

# NVidia dataset: 72 hrs of video => 72*60*60*30 = 7,776,000 images
# Nvidia blog: https://devblogs.nvidia.com/deep-learning-self-driving-cars/


# Our Dataset: https://github.com/SullyChen/Autopilot-TensorFlow [https://drive.
# Size: 25 minutes = 25*60*30 = 45,000 images ~ 2.3 GB


# If you want to try on a slightly large dataset: 70 minutes of data ~ 223GB
# Refer: https://medium.com/udacity/open-sourcing-223gb-of-mountain-view-driving
# Format: Image, latitude, longitude, gear, brake, throttle, steering angles and



# Additional Installations:
# pip3 install h5py


# AWS: https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learn

# Youtube:https://www.youtube.com/watch?v=qhUvQiKec2U
# Further reading and extensions: https://medium.com/udacity/teaching-a-machine-
# More data: https://medium.com/udacity/open-sourcing-223gb-of-mountain-view-dri
```

Train Test Split (70:30)

In [1]:

```
# read images and steering angles from driving_dataset folder

from __future__ import division

import os
import numpy as np
import random

from scipy import pi
from itertools import islice



DATA_FOLDER = './driving_dataset/' # change this to your folder
TRAIN_FILE = os.path.join(DATA_FOLDER, 'data.txt')


split =0.7
X = []
y = []
with open(TRAIN_FILE) as fp:
    for line in fp:
        path, angle = line.strip().split()
        full_path = os.path.join(DATA_FOLDER, path)
        X.append(full_path)

        # converting angle from degrees to radians
        y.append(float(angle) * pi / 180 )


y = np.array(y)
print("Completed processing data.txt")

split_index = int(len(y)*0.7)

train_y = y[:split_index]
test_y = y[split_index:]
```

```
Completed processing data.txt
```

In [9]:

```
len(X)
```

Out[9]:

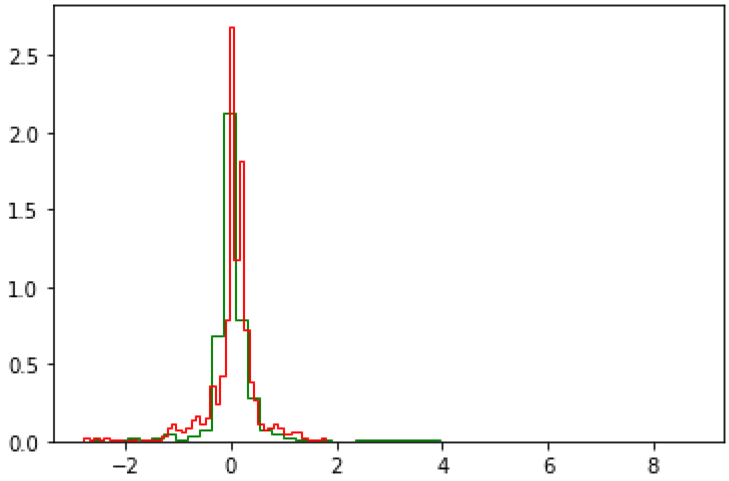
```
45406
```

In [3]:

```
import numpy;

# PDF of train and test 'y' values.
import matplotlib.pyplot as plt
plt.hist(train_y, bins=50, normed=1, color='green', histtype ='step');
plt.hist(test_y, bins=50, normed=1, color='red', histtype ='step');
plt.show()
```

```
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: MatplotlibDeprecationWa
rning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1.
Use 'density' instead.
  """
C:\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: MatplotlibDeprecationWa
rning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1.
Use 'density' instead.
```



In [4]:

```
#Model 0: Base line Model: y_test_pred = mean(y_train_i)
train_mean_y = np.mean(train_y)

print('Test_MSE(MEAN):%f' % np.mean(np.square(test_y-train_mean_y)) )

print('Test_MSE(ZERO):%f' % np.mean(np.square(test_y-0.0)) )
```

```
Test_MSE(MEAN):0.241561
Test_MSE(ZERO):0.241107
```

In [11]:

```
%%time
a=0
print(a)
```

```
0
Wall time: 0 ns
```

In [15]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
Time start = 11:30 AM 29/9/2019
```

Building the model with

- AdamOptimizer(1e-4)
- DropOut/keep_prob: 0.5
- Activation Function = tf.multiply((tf.matmul(h_fc4_drop, W_fc5) + b_fc5), 2)

In [16]:

```
%%time
import tensorflow as tf
from tensorflow.core.protobuf import saver_pb2
import driving_data
import model


LOGDIR = './My Final Save'

sess = tf.InteractiveSession()


L2NormConst = 0.001

train_vars = tf.trainable_variables()

loss = tf.reduce_mean(tf.square(tf.subtract(model.y_, model.y))) + tf.add_n([tf.
train_step = tf.train.AdamOptimizer(1e-4).minimize(loss)
sess.run(tf.initialize_all_variables())

# create a summary to monitor cost tensor
tf.summary.scalar("loss", loss)
# merge all summaries into a single op
merged_summary_op =  tf.summary.merge_all()

saver = tf.train.Saver(write_version = saver_pb2.SaverDef.V1)

# op to write logs to Tensorboard
logs_path = './logs'
summary_writer = tf.summary.FileWriter(logs_path, graph=tf.get_default_graph())

epochs = 30
batch_size = 100

# train over the dataset about 30 times
for epoch in range(epochs):
  for i in range(int(driving_data.num_images/batch_size)):
    xs, ys = driving_data.LoadTrainBatch(batch_size)
    train_step.run(feed_dict={model.x: xs, model.y_: ys, model.keep_prob: 0.5})
    if i % 10 == 0:
      xs, ys = driving_data.LoadValBatch(batch_size)
      loss_value = loss.eval(feed_dict={model.x:xs, model.y_: ys, model.keep_pro
      print("Epoch: %d, Step: %d, Loss: %g" % (epoch, epoch * batch_size + i, lo

    # write logs at every iteration
    summary = merged_summary_op.eval(feed_dict={model.x:xs, model.y_: ys, model.
    summary_writer.add_summary(summary, epoch * driving_data.num_images/batch_si

    if i % batch_size == 0:
      if not os.path.exists(LOGDIR):
        os.makedirs(LOGDIR)
      checkpoint_path = os.path.join(LOGDIR, "model.ckpt")
      filename = saver.save(sess, checkpoint_path)
  print("Model saved in file: %s" % filename)
print("----------->****Shritam Kumar Mund****<------------------")
print("Visit https://ishritam.ml for more detail about me.")
print("Run the command line:\n" \
          "--> tensorboard --logdir=./logs " \
          "\nThen open http://0.0.0.0:6006/ into your web browser")
```

```
WARNING:tensorflow:now on by default.

WARNING:tensorflow:now on by default.

WARNING:tensorflow:*******************************************************

WARNING:tensorflow:*******************************************************

Epoch: 29, Step: 3310, Loss: 0.18252
Epoch: 29, Step: 3320, Loss: 0.157792
Epoch: 29, Step: 3330, Loss: 0.259207
Epoch: 29, Step: 3340, Loss: 0.484205
Epoch: 29, Step: 3350, Loss: 0.259625
Model saved in file: ./My Final Save\model.ckpt
----------->****Shritam Kumar Mund****<------------------
Visit https://ishritam.ml (https://ishritam.ml) for more detail about me.
Run the command line:
--> tensorboard --logdir=./logs
Then open http://0.0.0.0:6006/ (http://0.0.0.0:6006/) into your web browser
Wall time: 17h 59min 48s
```

In [ ]: