In [1]:
```
from __future__ import division

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

import os
import random
import cv2
import math
import numpy as np
from scipy import pi
import scipy
import scipy.misc

from subprocess import call
from datetime import datetime
from itertools import islice
import matplotlib.pyplot as plt
import tensorflow as tf
```

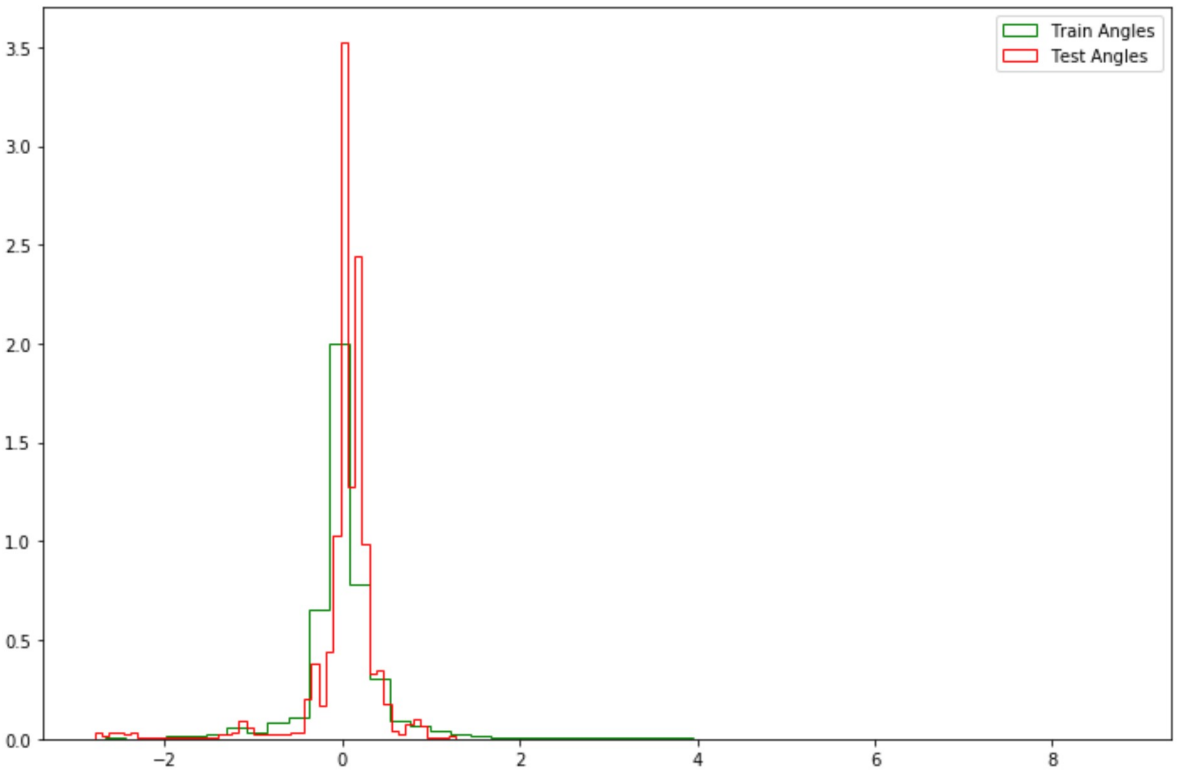
EDA

In [2]:
```
image_data = []
angle_data = []

# Get number of images
num_images = 0

# Number of images for training
num_train_images = 0

# Number of images for testing
num_test_images = 0

def load_dataset():
    # Read data.txt
    with open("driving_dataset\data.txt") as fp:
        for line in fp:
            image_data.append("driving_dataset/" + line.split()[0])

            # the paper by Nvidia uses the inverse of the turning radius,
            # but steering wheel angle is proportional to the inverse of turning ra
dius
            # so the steering wheel angle in radians is used as the output
            angle_data.append(float(line.split()[1]) * scipy.pi / 180)

def split_dataset(train_split,test_split):
    images_to_train = image_data[:int(len(image_data) * train_split)]
    angles_to_train = angle_data[:int(len(image_data) * train_split)]

    images_to_test = image_data[-int(len(image_data) * test_split):]
    angles_to_test = angle_data[-int(len(image_data) * test_split):]

    return images_to_train,angles_to_train,images_to_test,angles_to_test
```

In [3]:
```
load_dataset()

# Split dataset
images_to_train,angles_to_train,images_to_test,angles_to_test = split_dataset(0.8,0
.2)

num_images = len(image_data)
print("Total number of images: ",num_images)

num_train_images = len(images_to_train)
print("Total number of images for training: ",num_train_images)

num_test_images = len(images_to_test)
print("Total number of images for testing: ",num_test_images)
```

```
Total number of images:  45406
Total number of images for training:  36324
Total number of images for testing:  9081
```

In [4]:
```
# PDF of train and test angle values.
plt.figure(figsize=(12,8))
plt.hist(angles_to_train, bins=50, density=1, color='green', histtype ='step',label
="Train Angles")
plt.hist(angles_to_test, bins=50, density=1, color='red', histtype ='step',label="T
est Angles")
plt.legend()
plt.show()
```



Base line Model: y_test_pred = mean(y_train_i)

In [5]:
```
train_mean_angle = np.mean(angles_to_train)

print('Test_MSE(MEAN):%f' % np.mean(np.square(angles_to_test - train_mean_angle)))
```

```
Test_MSE(MEAN):0.191142
```

In [6]:
```
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)

def conv2d(x, W, stride):
    return tf.nn.conv2d(x, W, strides=[1, stride, stride, 1], padding='VALID')
```

Configuration Used

Train/Test Split: 70:30 Dropout : 0.50 AdamOptimezer Value: 1e-3 Activation Function: linear

In [7]:
```
true_image_ln = tf.placeholder(tf.float32, shape=[None, 66, 200, 3],name="true_imag
e_ln")
true_angle_ln = tf.placeholder(tf.float32, shape=[None, 1],name="true_angle_ln")

x_image_ln = true_image_ln

#first convolutional layer
W_conv1_ln = weight_variable([5, 5, 3, 24])
b_conv1_ln = bias_variable([24])

h_conv1_ln = tf.nn.relu(conv2d(x_image_ln, W_conv1_ln, 2) + b_conv1_ln)

#second convolutional layer
W_conv2_ln = weight_variable([5, 5, 24, 36])
b_conv2_ln = bias_variable([36])

h_conv2_ln = tf.nn.relu(conv2d(h_conv1_ln, W_conv2_ln, 2) + b_conv2_ln)

#third convolutional layer
W_conv3_ln = weight_variable([5, 5, 36, 48])
b_conv3_ln = bias_variable([48])

h_conv3_ln = tf.nn.relu(conv2d(h_conv2_ln, W_conv3_ln, 2) + b_conv3_ln)

#fourth convolutional layer
W_conv4_ln = weight_variable([3, 3, 48, 64])
b_conv4_ln = bias_variable([64])

h_conv4_ln = tf.nn.relu(conv2d(h_conv3_ln, W_conv4_ln, 1) + b_conv4_ln)

#fifth convolutional layer
W_conv5_ln = weight_variable([3, 3, 64, 64])
b_conv5_ln = bias_variable([64])

h_conv5_ln = tf.nn.relu(conv2d(h_conv4_ln, W_conv5_ln, 1) + b_conv5_ln)

#FCL 1
W_fc1_ln = weight_variable([1152, 1164])
b_fc1_ln = bias_variable([1164])

h_conv5_flat_ln = tf.reshape(h_conv5_ln, [-1, 1152])
h_fc1_ln = tf.nn.relu(tf.matmul(h_conv5_flat_ln, W_fc1_ln) + b_fc1_ln)

keep_prob_ln = tf.placeholder(tf.float32,name="keep_prob_ln")
h_fc1_drop_ln = tf.nn.dropout(h_fc1_ln, keep_prob_ln)

#FCL 2
W_fc2_ln = weight_variable([1164, 100])
b_fc2_ln = bias_variable([100])

h_fc2_ln = tf.nn.relu(tf.matmul(h_fc1_drop_ln, W_fc2_ln) + b_fc2_ln)

h_fc2_drop_ln = tf.nn.dropout(h_fc2_ln, keep_prob_ln)

#FCL 3
W_fc3_ln = weight_variable([100, 50])
b_fc3_ln = bias_variable([50])

h_fc3_ln = tf.nn.relu(tf.matmul(h_fc2_drop_ln, W_fc3_ln) + b_fc3_ln)

h_fc3_drop_ln = tf.nn.dropout(h_fc3_ln, keep_prob_ln)

#FCL 3
W_fc4_ln = weight_variable([50, 10])
```

In [8]:
```
train_batch_pointer = 0
test_batch_pointer = 0

# Utility Functions
def LoadTrainBatch(batch_size):
    global train_batch_pointer
    x_out = []
    y_out = []
    for i in range(0, batch_size):
        x_out.append(scipy.misc.imresize(scipy.misc.imread(images_to_train[(train_b
atch_pointer + i) % num_train_images])[-150:],
                                         [66, 200]) / 255.0)
        y_out.append([angles_to_train[(train_batch_pointer + i) % num_train_images]
])
    train_batch_pointer += batch_size
    return x_out, y_out

def LoadTestBatch(batch_size):
    global test_batch_pointer
    x_out = []
    y_out = []
    for i in range(0, batch_size):
        x_out.append(scipy.misc.imresize(scipy.misc.imread(images_to_test[(test_bat
ch_pointer + i) % num_test_images])[-150:],
                                         [66, 200]) / 255.0)
        y_out.append([angles_to_test[(test_batch_pointer + i) % num_test_images]])
    test_batch_pointer += batch_size
    return x_out, y_out
```

Model with 'linear' activation function

In [9]:
```
images_to_train,angles_to_train,images_to_test,angles_to_test = split_dataset(0.7,0
.3)

num_images = len(image_data)
print("Total number of images: ",num_images)

num_train_images = len(images_to_train)
print("Total number of images for training: ",num_train_images)

num_test_images = len(images_to_test)
print("Total number of images for testing: ",num_test_images)

# Reset the pointers
train_batch_pointer = 0
test_batch_pointer = 0
```

```
Total number of images:  45406
Total number of images for training:  31784
Total number of images for testing:  13621
```

In [11]:
```
LOGDIR = './models/linear'

# Lets start the tensorflow session
sess = tf.InteractiveSession()
```

```
C:\Users\Sai charan\Anaconda3\envs\tfgpu\lib\site-packages\tensorflow\python\cli
ent\session.py:1645: UserWarning: An interactive session is already active. This
can cause out-of-memory errors in some cases. You must explicitly call `Interact
iveSession.close()` to release resources held by the other session(s).
  warnings.warn('An interactive session is already active. This can '
```

In [12]:
```
start = datetime.now()

print("Let the model learn itself...")
print()

L2NormConst = 0.001

train_vars = tf.trainable_variables()

loss = tf.reduce_mean(tf.square(tf.subtract(true_angle_ln, predicted_angle_ln))) +
tf.add_n([tf.nn.l2_loss(v) for v in train_vars]) * L2NormConst
train_step = tf.train.AdamOptimizer(1e-3).minimize(loss)

sess.run(tf.global_variables_initializer())

# 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()

# 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
previous_i = 0
previous_loss = 0
for epoch in range(epochs):
    for i in range(int(num_images/batch_size)):
        xs, ys = LoadTrainBatch(batch_size)
        train_step.run(feed_dict={true_image_ln: xs, true_angle_ln: ys, keep_prob_l
n: 0.50})
        if i % 10 == 0:
            xs, ys = LoadTestBatch(batch_size)
            loss_value = loss.eval(feed_dict={true_image_ln:xs, true_angle_ln: ys,
keep_prob_ln: 1.0})
            previous_loss = loss_value
            previous_i = i
            # print("Epoch: %d, Step: %d, Loss: %g" % (epoch, epoch * batch_size +
i, loss_value))

        # write logs at every iteration
        summary = merged_summary_op.eval(feed_dict={true_image_ln:xs, true_angle_ln
: ys, keep_prob_ln: 1.0})
        summary_writer.add_summary(summary, epoch * num_images/batch_size + i)

        if i % batch_size == 0:
            if not os.path.exists(LOGDIR):
                os.makedirs(LOGDIR)
            checkpoint_path = os.path.join(LOGDIR, "model_linear.ckpt")
            filename = saver.save(sess, checkpoint_path)
    print("Epoch: %d, Step: %d, Loss: %g" % (epoch, epoch * batch_size + previous_i
, previous_loss))
    print("Model saved in file: %s" % filename)
    print()

print("Run the command line:\n" \
          "--> tensorboard --logdir=./logs " \
```

```
Let the model learn itself...

Epoch: 0, Step: 450, Loss: 1.94947
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 1, Step: 550, Loss: 0.885368
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 2, Step: 650, Loss: 0.480319
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 3, Step: 750, Loss: 0.844324
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 4, Step: 850, Loss: 0.167221
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 5, Step: 950, Loss: 0.218046
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 6, Step: 1050, Loss: 0.135026
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 7, Step: 1150, Loss: 0.0505313
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 8, Step: 1250, Loss: 0.344266
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 9, Step: 1350, Loss: 0.0344594
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 10, Step: 1450, Loss: 0.0158085
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 11, Step: 1550, Loss: 0.186574
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 12, Step: 1650, Loss: 0.573643
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 13, Step: 1750, Loss: 0.00376124
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 14, Step: 1850, Loss: 0.21957
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 15, Step: 1950, Loss: 0.748492
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 16, Step: 2050, Loss: 0.00205246
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 17, Step: 2150, Loss: 0.0462229
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 18, Step: 2250, Loss: 0.0882451
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 19, Step: 2350, Loss: 0.00709648
Model saved in file: ./models/linear\model_linear.ckpt

Epoch: 20, Step: 2450, Loss: 0.0490244
Model saved in file: ./models/linear\model_linear.ckpt
```

In [13]:
```
sess.close()
```

Model Testing

To run model, open command prompt or terminal and type 'pyhton3 run_linear.py'

In [ ]:
```

```