In [18]:

```
import scipy.misc
import random
import imageio
from skimage.transform import resize
xs = []
ys = []

#points to the end of the last batch
train_batch_pointer = 0
val_batch_pointer = 0

#read data.txt
with open("driving_dataset/data.txt") as f:
    for line in f:
        xs.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 r
        #so the steering wheel angle in radians is used as the output
        ys.append(float(line.split()[1]) * scipy.pi / 180)

#get number of images
num_images = len(xs)


train_xs = xs[:int(len(xs) * 0.7)]
train_ys = ys[:int(len(xs) * 0.7)]

val_xs = xs[-int(len(xs) * 0.3):]
val_ys = ys[-int(len(xs) * 0.3):]

num_train_images = len(train_xs)
num_val_images = len(val_xs)

def LoadTrainBatch(batch_size):
    global train_batch_pointer
    x_out = []
    y_out = []
    for i in range(0, batch_size):
        image = full_image[-150:]/255.0
        image=resize(image,[66,200])
        x_out.append(resize(imread(train_xs[(train_batch_pointer + i) % num_t
        y_out.append([train_ys[(train_batch_pointer + i) % num_train_images]]
    train_batch_pointer += batch_size
    return x_out, y_out

def LoadValBatch(batch_size):
    global val_batch_pointer
    x_out = []
    y_out = []
    for i in range(0, batch_size):
        x_out.append(resize(imread(val_xs[(val_batch_pointer + i) % num_val_i
        y_out.append([val_ys[(val_batch_pointer + i) % num_val_images]])
    val_batch_pointer += batch_size
    return x_out, y_out
```

In [ ]:

```
import tensorflow as tf
import scipy

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')

x = tf.placeholder(tf.float32, shape=[None, 66, 200, 3])
y_ = tf.placeholder(tf.float32, shape=[None, 1])

x_image = x

#first convolutional layer
W_conv1 = weight_variable([5, 5, 3, 24])
b_conv1 = bias_variable([24])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1, 2) + b_conv1)

#second convolutional layer
W_conv2 = weight_variable([5, 5, 24, 36])
b_conv2 = bias_variable([36])

h_conv2 = tf.nn.relu(conv2d(h_conv1, W_conv2, 2) + b_conv2)

#third convolutional layer
W_conv3 = weight_variable([5, 5, 36, 48])
b_conv3 = bias_variable([48])

h_conv3 = tf.nn.relu(conv2d(h_conv2, W_conv3, 2) + b_conv3)

#fourth convolutional layer
W_conv4 = weight_variable([3, 3, 48, 64])
b_conv4 = bias_variable([64])

h_conv4 = tf.nn.relu(conv2d(h_conv3, W_conv4, 1) + b_conv4)

#fifth convolutional layer
W_conv5 = weight_variable([3, 3, 64, 64])
b_conv5 = bias_variable([64])

h_conv5 = tf.nn.relu(conv2d(h_conv4, W_conv5, 1) + b_conv5)

#FCL 1
W_fc1 = weight_variable([1152, 1164])
b_fc1 = bias_variable([1164])

h_conv5_flat = tf.reshape(h_conv5, [-1, 1152])
h_fc1 = tf.nn.relu(tf.matmul(h_conv5_flat, W_fc1) + b_fc1)
```

```
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

#FCL 2
W_fc2 = weight_variable([1164, 100])
b_fc2 = bias_variable([100])

h_fc2 = tf.nn.relu(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)

h_fc2_drop = tf.nn.dropout(h_fc2, keep_prob)

#FCL 3
W_fc3 = weight_variable([100, 50])
b_fc3 = bias_variable([50])

h_fc3 = tf.nn.relu(tf.matmul(h_fc2_drop, W_fc3) + b_fc3)

h_fc3_drop = tf.nn.dropout(h_fc3, keep_prob)

#FCL 3
W_fc4 = weight_variable([50, 10])
b_fc4 = bias_variable([10])

h_fc4 = tf.nn.relu(tf.matmul(h_fc3_drop, W_fc4) + b_fc4)

h_fc4_drop = tf.nn.dropout(h_fc4, keep_prob)

#Output
W_fc5 = weight_variable([10, 1])
b_fc5 = bias_variable([1])

y = tf.identity(tf.matmul(h_fc4_drop, W_fc5) + b_fc5) #scale the atan output
```

In [ ]:

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

LOGDIR = './save_identity_4_0.5'

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([
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 = './save_identity_4_0.5'
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_
      print("Epoch: %d, Step: %d, Loss: %g" % (epoch, epoch * batch_size + i,

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

    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("Run the command line:\n" \
          "--> tensorboard --logdir=./logs " \
          "\nThen open http://0.0.0.0:6006/ into your web browser")
```

In [1]:

```
#pip3 install opencv-python
import imageio
from skimage.transform import resize
import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import scipy.misc
import model
import cv2
from subprocess import call
import math
from matplotlib.pyplot import imread
from PIL import Image

sess = tf.InteractiveSession()
saver = tf.train.Saver()
saver.restore(sess, "save_identity_4_0.5/model.ckpt")

img = cv2.imread('steering_wheel_image.jpg',0)
rows,cols = img.shape

smoothed_angle = 0


#read data.txt
xs = []
ys = []
with open("driving_dataset/data.txt") as f:
    for line in f:
        xs.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 r
        #so the steering wheel angle in radians is used as the output
        ys.append(float(line.split()[1]) * scipy.pi / 180)

#get number of images
num_images = len(xs)


i = math.ceil(num_images*0.8)
print("Starting frameofvideo:" +str(i))

while(cv2.waitKey(10) != ord('q')):
    full_image =imread("driving_dataset/" + str(i) + ".jpg")
    image = full_image[-150:]/255.0
    image=resize(image,[66,200])
    degrees = model.y.eval(feed_dict={model.x: [image], model.keep_prob: 1.0}
    #call("clear")
    #print("Predicted Steering angle: " + str(degrees))
    print("Steering angle: " + str(degrees) + " (pred)\t" + str(ys[i]*180/sci
    cv2.imshow("frame", cv2.cvtColor(full_image, cv2.COLOR_RGB2BGR))
    #make smooth angle transitions by turning the steering wheel based on the
    #and the predicted angle
    smoothed_angle += 0.2 * pow(abs((degrees - smoothed_angle)), 2.0 / 3.0) *
    M = cv2.getRotationMatrix2D((cols/2,rows/2),-smoothed_angle,1)
    dst = cv2.warpAffine(img,M,(cols,rows))
```

```
    cv2.imshow("steering wheel", dst)
    i += 1

cv2.destroyAllWindows()
```

```
WARNING:tensorflow:From C:\Users\nnagari\AppData\Local\Continuum\anaconda
3\lib\site-packages\tensorflow_core\python\compat\v2_compat.py:65: disabl
e_resource_variables (from tensorflow.python.ops.variable_scope) is depre
cated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
WARNING:tensorflow:From C:\Users\nnagari\Desktop\ML\Autopilot-TensorFlow-
master\Autopilot-TensorFlow-master\model.py:63: calling dropout (from ten
sorflow.python.ops.nn_ops) with keep_prob is deprecated and will be remov
ed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
INFO:tensorflow:Restoring parameters from save_identity_4_0.5/model.ckpt
Starting frameofvideo:36325
Steering angle: 5.401364395178035 (pred)        -10.79 (actual)
Steering angle: 5.950197842632063 (pred)        -10.08 (actual)
Steering angle: 6.092030989615609 (pred)        -9.380000000000003 (actua
l)
```

In [9]:

```
import prettytable

table=prettytable.PrettyTable()

table.field_names=['Model activation','keepprob','Loss']

table.add_row(['identity','1e-4','0.4013'])
#table.add_row(['identity','1e-3','0.4425',""])



print(table)
```

```
+------------------+----------+--------+
| Model activation | keepprob |  Loss  |
+------------------+----------+--------+
|     identity     |   1e-4   | 0.4013 |
+------------------+----------+--------+
```

1. This model comparatively doing good when we compared below model but sometimes it will do mistakes but it is not best compare to base model.

In [11]:
```
import prettytable

table=prettytable.PrettyTable()

table.field_names=['Model activation','keepprob','Loss']

#table.add_row(['identity','1e-4','0.4013'])
table.add_row(['identity','1e-3','0.4425'])



print(table)
```

```
+------------------+----------+--------+
| Model activation | keepprob |  Loss  |
+------------------+----------+--------+
|     identity     |   1e-3   | 0.4425 |
+------------------+----------+--------+
```

1. "Above model is not performing well compared to base model, in some situation model supposed to predict the -VE steering angle but it is predicting the +VE steering angle."