In [0]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call d
rive.mount("/content/drive", force_remount=True).
```

In [1]:
```
!pip3 install patool
import patoolib
patoolib.extract_archive('/content/drive/My Drive/Autopilot-TensorFlow-master.ra
```

```
Collecting patool
  Downloading https://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d
8110964320ab4851134a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.whl (htt
ps://files.pythonhosted.org/packages/43/94/52243ddff508780dd2d8110964320ab48511
34a55ab102285b46e740f76a/patool-1.12-py2.py3-none-any.whl) (77kB)
     |████████████████████████████████| 81kB 2.5MB/s eta 0:00:011
Installing collected packages: patool
Successfully installed patool-1.12
patool: Extracting /content/drive/My Drive/Autopilot-TensorFlow-master.rar ...
patool: running /usr/bin/unrar x -- "/content/drive/My Drive/Autopilot-TensorFl
ow-master.rar"
patool:     with cwd='./Unpack_gce_1zzn'
patool: ... /content/drive/My Drive/Autopilot-TensorFlow-master.rar extracted t
o `Autopilot-TensorFlow-master'.
```

Out[1]:
```
'Autopilot-TensorFlow-master'
```

Lading of the Driving_data

In [0]:
```
#  !pip install scipy
import scipy.misc
import random

xs = []
ys = []

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

#read data.txt
with open("/content/Autopilot-TensorFlow-master/Autopilot-TensorFlow-master/drivi
    for line in f:
        xs.append("/content/Autopilot-TensorFlow-master/Autopilot-TensorFlow-mast
        #the paper by Nvidia uses the inverse of the turning radius,
        #but steering wheel angle is proportional to the inverse of turning radiu
        #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)

import imageio
import skimage.transform

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):
        x_out.append(skimage.transform.resize(imageio.imread(train_xs[(train_batc
        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(skimage.transform.resize(imageio.imread(val_xs[(val_batch_po
        y_out.append([val_ys[(val_batch_pointer + i) % num_val_images]])
    val_batch_pointer += batch_size
    return x_out, y_out
```

RNN_CNN---Model

In [3]:
```
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.compat.v1.placeholder(tf.float32, shape=[None, 66, 200, 3])
y_ = tf.compat.v1.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.multiply(tf.matmul(h_fc4_drop, W_fc5) + b_fc5, 2)
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

```
WARNING:tensorflow:From <ipython-input-3-8a4f20c9206a>:58: calling dropout (fro
m tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be remove
d in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
```

Training and Evaluvation of Loss using Adam optimizer(1e-3)

In [4]:
```
import os
import tensorflow as tf
from tensorflow.core.protobuf import saver_pb2

# import driving_data
# import model

LOGDIR = './save'

sess = tf.InteractiveSession()

L2NormConst = 0.001

train_vars = tf.trainable_variables()

loss = tf.reduce_mean(tf.square(tf.subtract(y_, y))) + tf.add_n([tf.nn.l2_loss(v
train_step = tf.train.AdamOptimizer(1e-3).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(num_images/batch_size)):
    xs, ys = LoadTrainBatch(batch_size)
    train_step.run(feed_dict={x: xs, y_: ys, keep_prob: 0.5})
    if i % 10 == 0:
      xs, ys = LoadValBatch(batch_size)
      loss_value = loss.eval(feed_dict={x:xs, y_: ys, keep_prob: 1.0})
      print("Epoch: %d, Step: %d, Loss: %g" % (epoch, epoch * batch_size + i, los

    # write logs at every iteration
    summary = merged_summary_op.eval(feed_dict={x:xs, y_: ys, keep_prob: 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.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")
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_cor
e/python/util/tf_should_use.py:198: initialize_all_variables (from tensorflo
w.python.ops.variables) is deprecated and will be removed after 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
Epoch: 0, Step: 0, Loss: 6.62516
Epoch: 0, Step: 10, Loss: 5.54357
Epoch: 0, Step: 20, Loss: 5.04342
Epoch: 0, Step: 30, Loss: 4.64451
Epoch: 0, Step: 40, Loss: 4.4996
Epoch: 0, Step: 50, Loss: 3.98863
Epoch: 0, Step: 60, Loss: 3.94773
Epoch: 0, Step: 70, Loss: 3.96057
Epoch: 0, Step: 80, Loss: 3.6136
Epoch: 0, Step: 90, Loss: 3.17738
Epoch: 0, Step: 100, Loss: 3.01685
Epoch: 0, Step: 110, Loss: 2.80845
Epoch: 0, Step: 120, Loss: 2.69694
Epoch: 0, Step: 130, Loss: 2.95446
```