Self Driving Car Assignment Using keras

Loading Dataset

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

import os
import numpy as np
import random
from scipy import pi
import imageio
import cv2
from itertools import islice


LIMIT = None

DATA_FOLDER = 'driving_dataset'
TRAIN_FILE = os.path.join(DATA_FOLDER, 'data.txt')

# the expected dimension is (tf ordering)
# (None, 66, 200, 3)
def return_data(split=.7):

    X = []
    y = []

    with open(TRAIN_FILE) as fp:
        for line in islice(fp, LIMIT):
            path, angle = line.strip().split()
            full_path = os.path.join(DATA_FOLDER, path)
            X.append(full_path)
            # using angles from -pi to pi to avoid rescaling th
e atan in the network
            y.append(float(angle) * pi / 180 )
    y = np.array(y)


    #images = np.array([np.float32(imageio.imread(im))/255  for
im in X])
    images = np.array([np.float32(cv2.resize(imageio.imread(im,
pilmode="RGB")[-150:], dsize=(200,66)))/255  for im in X])
    split_index = int(split * len(X))

    train_X = images[:split_index]
    train_y = y[:split_index]
    test_X = images[split_index:]
    test_y = y[split_index:]

    return np.array(train_X), np.array(train_y), np.array(test_
X), np.array(test_y)
```

Model Arcitecture

In [2]:
```
from keras.layers import Convolution2D, Input
from keras.layers.core import Dense, Flatten, Lambda, Activatio
n, Dropout
from keras.models import Model, Sequential
from keras.optimizers import SGD
import keras
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
RemoteMonitor, EarlyStopping
from keras.layers import Conv2D,MaxPooling2D

import tensorflow as tf
"""
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.9
sess = InteractiveSession(config=config)"""

from keras import backend as K
K.common.image_dim_ordering()


# fina NVIDIA Architecture Model
# reference: https://github.com/hdmetor/Nvidia-SelfDriving

def NVIDA( lr_rate, d_rate, verbose=True ):


    nvidia_model= Sequential()
    nvidia_model.add(Convolution2D(24, kernel_size=(5, 5),paddi
ng='valid', activation='relu', strides=(2, 2), name='conv_1',in
put_shape=(66,200, 3) ))
    nvidia_model.add(Convolution2D(36, kernel_size=(5, 5),paddi
ng='valid', activation='relu', strides=(2, 2), name='conv_2' ))
    nvidia_model.add(Convolution2D(48, kernel_size=(5, 5),paddi
ng='valid', activation='relu', strides=(2, 2), name='conv_3' ))
    nvidia_model.add(Convolution2D(64, kernel_size=(3, 3),paddi
ng='valid', activation='relu', strides=(1, 1), name='conv_4'))
    nvidia_model.add(Convolution2D(64, kernel_size=(3, 3),paddi
ng='valid', activation='relu', strides=(1, 1), name='conv_5'))

    nvidia_model.add(Flatten())

    nvidia_model.add(Dense(1164, activation="relu"))
    nvidia_model.add(Dropout(rate=d_rate))

    nvidia_model.add(Dense(100, activation="relu"))
    nvidia_model.add(Dropout(rate=d_rate))

    nvidia_model.add(Dense(50, activation="relu"))
    nvidia_model.add(Dropout(rate=d_rate))

    nvidia_model.add(Dense(10, activation="relu"))
    nvidia_model.add(Dropout(rate=d_rate))

    nvidia_model.add(Dense(1))
    nvidia_model.add(Lambda(lambda x: keras.activations.tanh(x
)))

    #optimiser inisilisation
```

```
    ada=keras.optimizers.adam(lr_rate)

    nvidia_model.compile(optimizer=ada,loss='mse')
    if verbose:
        nvidia_model.summary()

    return nvidia_model
```

```
Using TensorFlow backend.
```

```
'tf'
```

Loading and Traning and Hyperparameter Tuning

In [4]:
```
import time
import json
import numpy as np
import pickle

print('Loading data...')
train_x, train_y, test_x, test_y = return_data(split=0.7)
print('Done')
```

```
Loading data...
Done
```

In [5]:
```
print(f"train shape: {train_x.shape}")
print(f"test shape: {test_x.shape}")
```

```
train shape: (31784, 66, 200, 3)
test shape: (13622, 66, 200, 3)
```

In [20]:
```
# Hyperparameter Tuning

#checkpoint callback
checkpointer = ModelCheckpoint(
    filepath="{epoch:02d}-{val_loss:.12f}.hdf5",
    verbose=1,
    save_best_only=True)

# Early Stopper callback
early_stopper = EarlyStopping(monitor='val_loss',min_delta=1e-4
,patience=10,verbose=0,
                              mode='auto',baseline=None,restore
_best_weights=True)

# reduce_learning_rate plateau callback
lr_plateau = ReduceLROnPlateau(monitor='val_loss', factor=0.1,
patience=10, min_lr=0.000001, verbose=0, mode=min)


def SDC(train_x, train_y, test_x, test_y, params):

    nvidia_model= Sequential()
```

```
    nvidia_model.add(Convolution2D(24, kernel_size=(5, 5),paddi
ng='valid', activation='relu', strides=(2, 2), name='conv_1',in
put_shape=(66,200, 3) ))
    nvidia_model.add(Convolution2D(36, kernel_size=(5, 5),paddi
ng='valid', activation='relu', strides=(2, 2), name='conv_2' ))
    nvidia_model.add(Convolution2D(48, kernel_size=(5, 5),paddi
ng='valid', activation='relu', strides=(2, 2), name='conv_3' ))
    nvidia_model.add(Convolution2D(64, kernel_size=(3, 3),paddi
ng='valid', activation='relu', strides=(1, 1), name='conv_4'))
    nvidia_model.add(Convolution2D(64, kernel_size=(3, 3),paddi
ng='valid', activation='relu', strides=(1, 1), name='conv_5'))

    nvidia_model.add(Flatten())

    nvidia_model.add(Dense(1164, activation="relu"))
    nvidia_model.add(Dropout(rate=params["d_rate"]))

    nvidia_model.add(Dense(100, activation="relu"))
    nvidia_model.add(Dropout(rate=params["d_rate"]))

    nvidia_model.add(Dense(50, activation="relu"))
    nvidia_model.add(Dropout(rate=params["d_rate"]))

    nvidia_model.add(Dense(10, activation="relu"))
    nvidia_model.add(Dropout(rate=params["d_rate"]))

    nvidia_model.add(Dense(1))
    nvidia_model.add(Lambda(lambda x: keras.activations.tanh(x
)))

    #optimiser inisilisation
    ada=keras.optimizers.adam(learning_rate=params["lr"])

    nvidia_model.compile(optimizer=ada,loss='mse')

    history = nvidia_model.fit(train_x, train_y,
                        validation_data=[test_x, test_y],
                        batch_size=params['batch_size'],
                        epochs=params['epochs'],
                        verbose=0,
                        callbacks=[early_stopper,lr_plateau])

    # finally we have to make sure that history object and mode
l are returned
    return history, nvidia_model
```

In [21]:
```
# making dict of hyperparameters
p = {'batch_size': (25,50,100,125),
     'epochs': [100],
     'd_rate':[0.5],
     "lr":[0.001,0.0001]}

p
```

```
{'batch_size': (25, 50, 100, 125),
 'epochs': [100],
 'd_rate': [0.5],
 'lr': [0.001, 0.0001]}
```

In [22]:
```
# hypertuning parameter using takos
```

```python
import talos

t = talos.Scan(x=train_x,
            y=train_y,
            model=SDC,
            params=p,
            experiment_name='sdc_fnal')
```

```
  0%|
| 0/50 [00:00<?, ?it/s]

  2%|█
| 1/50 [01:45<1:25:50, 105.11s/it]

  4%|██
| 2/50 [09:41<2:53:10, 216.46s/it]

  6%|███
| 3/50 [11:21<2:22:18, 181.68s/it]

  8%|████
| 4/50 [22:28<4:10:51, 327.21s/it]

 10%|█████
| 5/50 [24:16<3:16:02, 261.39s/it]

 12%|██████
| 6/50 [33:46<4:19:37, 354.04s/it]
```

```
14%|
| 7/50 [35:26<3:18:57, 277.62s/it]

16%|
| 8/50 [48:21<4:58:56, 427.05s/it]

18%|
| 9/50 [49:57<3:43:56, 327.72s/it]

20%|
| 10/50 [57:26<4:02:41, 364.05s/it]

22%|
| 11/50 [59:10<3:06:00, 286.16s/it]

24%|
| 12/50 [1:07:54<3:46:14, 357.24s/it]

26%|
| 13/50 [1:09:28<2:51:39, 278.36s/it]

28%|
| 14/50 [1:14:57<2:56:05, 293.49s/it]

30%|
| 15/50 [1:16:37<2:17:26, 235.61s/it]
```

32%|██████████████████████████
| 16/50 [1:26:52<3:17:56, 349.31s/it]

34%|███████████████████████████
| 17/50 [1:28:24<2:29:42, 272.19s/it]

36%|█████████████████████████████
| 18/50 [1:41:28<3:47:04, 425.77s/it]

38%|██████████████████████████████
| 19/50 [1:42:59<2:48:02, 325.23s/it]

40%|████████████████████████████████
| 20/50 [1:51:54<3:14:04, 388.16s/it]

42%|█████████████████████████████████
| 21/50 [1:53:22<2:24:08, 298.23s/it]

44%|███████████████████████████████████
| 22/50 [2:02:41<2:55:35, 376.28s/it]

46%|█████████████████████████████████████
| 23/50 [2:04:48<2:15:46, 301.74s/it]

```
 48%|
| 24/50 [2:15:40<2:56:13, 406.68s/it]

 50%|
| 25/50 [2:17:13<2:10:16, 312.66s/it]

 52%|
| 26/50 [2:26:22<2:33:25, 383.55s/it]

 54%|
| 27/50 [2:27:59<1:54:05, 297.65s/it]

 56%|
| 28/50 [2:39:34<2:32:50, 416.85s/it]

 58%|
| 29/50 [2:41:22<1:53:25, 324.05s/it]

 60%|
| 30/50 [2:48:09<1:56:19, 349.00s/it]

 62%|
| 31/50 [2:49:49<1:26:49, 274.16s/it]

 64%|
| 32/50 [2:57:09<1:37:12, 324.04s/it]
```

```
 66%|████████████████████████████████████████████████████
█                            | 33/50 [2:58:56<1:13:22, 25
8.98s/it]

 68%|█████████████████████████████████████████████████████
█▍                          | 34/50 [3:09:06<1:37:08, 36
4.30s/it]

 70%|█████████████████████████████████████████████████████
██▌                         | 35/50 [3:10:30<1:10:02, 28
0.14s/it]

 72%|█████████████████████████████████████████████████████
████▋                       | 36/50 [3:18:20<1:18:41, 33
7.22s/it]

 74%|█████████████████████████████████████████████████████
███████▊                      | 37/50 [3:19:51<57:00, 26
3.08s/it]

 76%|█████████████████████████████████████████████████████
████████                    | 38/50 [3:31:20<1:18:12, 391.
02s/it]

 78%|█████████████████████████████████████████████████████
███████████                   | 39/50 [3:32:41<54:38, 298.
02s/it]

 80%|█████████████████████████████████████████████████████
```

[redacted] | 40/50 [3:43:15<1:06:26, 398.
69s/it]

82%|[redacted] | 41/50 [3:44:36<45:30, 30
3.44s/it]

84%|[redacted] | 42/50 [3:55:08<53:36, 40
2.05s/it]

86%|[redacted] | 43/50 [3:56:36<35:55, 30
7.95s/it]

88%|[redacted] | 44/50 [4:07:51<41:47, 417.
90s/it]

90%|[redacted] | 45/50 [4:09:27<26:46, 321.
30s/it]

92%|[redacted] | 46/50 [4:17:33<24:42, 37
0.69s/it]

94%|[redacted] | 47/50 [4:18:58<14:15, 28
5.17s/it]

```
 96%|████████████████████████████████████████████████▏ | 48/50 [4:27:33<11:47, 353.93s/it]
```

```
 98%|█████████████████████████████████████████████████  | 49/50 [4:28:59<04:33, 273.53s/it]
```

```
100%|██████████████████████████████████████████████████| 50/50 [4:37:15<00:00, 332.72s/it]
```

In [27]:
```python
result = t.data
result.head()
```

	round_epochs	val_loss	loss	lr	batch_size	d_rate	epo
0	11	0.319554	0.309572	0.0010	25	0.5	100
1	52	0.122518	0.127042	0.0001	25	0.5	100
2	11	0.319588	0.309542	0.0010	26	0.5	100
3	74	0.121853	0.121453	0.0001	26	0.5	100
4	12	0.319623	0.309552	0.0010	27	0.5	100

In [26]:
```python
# optimal hyperparameter
result[result.val_loss==result.val_loss.min()]
```

	round_epochs	val_loss	loss	lr	batch_size	d_rate	ep
39	87	0.121049	0.125895	0.0001	44	0.5	10

Traning using optimal hyper parameter

In [28]:
```python
# Traning using optimal hyper parameter

#checkpoint callback
checkpointer = ModelCheckpoint(
    filepath="{epoch:02d}-{val_loss:.12f}.hdf5",
    verbose=1,
    save_best_only=True)

# Early Stopper callback
early_stopper = EarlyStopping(monitor='val_loss',min_delta=1e-4
,patience=10,verbose=2,
                              mode='auto',baseline=None,restore
```

```python
_best_weights=True)

# reduce_learning_rate plateau callback
lr_plateau = ReduceLROnPlateau(monitor='val_loss', factor=0.1,
patience=10, min_lr=0.000001, verbose=1, mode=min)


# optimal hyperparameter
epochs = 100
batch_size = 44
lr=0.0001
d=0.5

print('Loading model')
nvidia = NVIDA(lr_rate=lr, d_rate=d)
print('Starting training')
history = nvidia.fit(train_x, train_y,
                     validation_data=(test_x, test_y),
                     nb_epoch=epochs,
                     batch_size=batch_size,
                     verbose=2,
                     callbacks=[checkpointer])


with open( f'history_{time.time()}.pkl' , 'wb') as fp:
    # dict with np array as items are not serializable
    pickle.dump(dict((k, np.array(v).tolist()) for k, v in hist
ory.history.items()), fp)
print('Done')
```

```
Loading model
Model: "sequential_1"
_________________________________________________________________
_________
Layer (type)                 Output Shape              P
aram #
=================================================================
=========
conv_1 (Conv2D)              (None, 31, 98, 24)        1
824
_________________________________________________________________
_________
conv_2 (Conv2D)              (None, 14, 47, 36)        2
1636
_________________________________________________________________
_________
conv_3 (Conv2D)              (None, 5, 22, 48)         4
3248
_________________________________________________________________
_________
conv_4 (Conv2D)              (None, 3, 20, 64)         2
7712
_________________________________________________________________
_________
conv_5 (Conv2D)              (None, 1, 18, 64)         3
6928
_________________________________________________________________
```

```
_________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
_________
dense_1 (Dense)              (None, 1164)              1
342092
_________________________________________________________________
_________
dropout_1 (Dropout)          (None, 1164)              0
_________________________________________________________________
_________
dense_2 (Dense)              (None, 100)               1
16500
_________________________________________________________________
_________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
_________
dense_3 (Dense)              (None, 50)                5
050
_________________________________________________________________
_________
dropout_3 (Dropout)          (None, 50)                0
_________________________________________________________________
_________
dense_4 (Dense)              (None, 10)                5
10
_________________________________________________________________
_________
dropout_4 (Dropout)          (None, 10)                0
_________________________________________________________________
_________
dense_5 (Dense)              (None, 1)                 1
1
_________________________________________________________________
_________
lambda_1 (Lambda)            (None, 1)                 0
=================================================================
=========
Total params: 1,595,511
Trainable params: 1,595,511
Non-trainable params: 0
_________________________________________________________________
_________
Starting training
Train on 31784 samples, validate on 13622 samples
Epoch 1/100
 - 15s - loss: 0.3129 - val_loss: 0.2402

Epoch 00001: val_loss improved from inf to 0.24019, savi
ng model to 01-0.240193529073.hdf5
Epoch 2/100
```

```
 - 10s - loss: 0.3081 - val_loss: 0.2352

Epoch 00002: val_loss improved from 0.24019 to 0.23518,
saving model to 02-0.235184571388.hdf5
Epoch 3/100
 - 10s - loss: 0.2909 - val_loss: 0.2353

Epoch 00003: val_loss did not improve from 0.23518
Epoch 4/100
 - 10s - loss: 0.2593 - val_loss: 0.2522

Epoch 00004: val_loss did not improve from 0.23518
Epoch 5/100
 - 10s - loss: 0.2199 - val_loss: 0.2094

Epoch 00005: val_loss improved from 0.23518 to 0.20940,
saving model to 05-0.209404710160.hdf5
Epoch 6/100
 - 10s - loss: 0.1965 - val_loss: 0.1984

Epoch 00006: val_loss improved from 0.20940 to 0.19845,
saving model to 06-0.198446688075.hdf5
Epoch 7/100
 - 10s - loss: 0.1771 - val_loss: 0.2066

Epoch 00007: val_loss did not improve from 0.19845
Epoch 8/100
 - 10s - loss: 0.1660 - val_loss: 0.2009

Epoch 00008: val_loss did not improve from 0.19845
Epoch 9/100
 - 10s - loss: 0.1595 - val_loss: 0.1861

Epoch 00009: val_loss improved from 0.19845 to 0.18613,
saving model to 09-0.186132975735.hdf5
Epoch 10/100
 - 10s - loss: 0.1499 - val_loss: 0.2067

Epoch 00010: val_loss did not improve from 0.18613
Epoch 11/100
 - 10s - loss: 0.1459 - val_loss: 0.1854

Epoch 00011: val_loss improved from 0.18613 to 0.18538,
saving model to 11-0.185376542353.hdf5
Epoch 12/100
 - 10s - loss: 0.1417 - val_loss: 0.1766

Epoch 00012: val_loss improved from 0.18538 to 0.17664,
saving model to 12-0.176639022486.hdf5
Epoch 13/100
 - 10s - loss: 0.1401 - val_loss: 0.1903
```

```
Epoch 00013: val_loss did not improve from 0.17664
Epoch 14/100
 - 10s - loss: 0.1392 - val_loss: 0.2066

Epoch 00014: val_loss did not improve from 0.17664
Epoch 15/100
 - 10s - loss: 0.1376 - val_loss: 0.2198

Epoch 00015: val_loss did not improve from 0.17664
Epoch 16/100
 - 10s - loss: 0.1351 - val_loss: 0.2273

Epoch 00016: val_loss did not improve from 0.17664
Epoch 17/100
 - 10s - loss: 0.1339 - val_loss: 0.1885

Epoch 00017: val_loss did not improve from 0.17664
Epoch 18/100
 - 10s - loss: 0.1357 - val_loss: 0.2245

Epoch 00018: val_loss did not improve from 0.17664
Epoch 19/100
 - 10s - loss: 0.1334 - val_loss: 0.2149

Epoch 00019: val_loss did not improve from 0.17664
Epoch 20/100
 - 10s - loss: 0.1310 - val_loss: 0.2186

Epoch 00020: val_loss did not improve from 0.17664
Epoch 21/100
 - 10s - loss: 0.1332 - val_loss: 0.2124

Epoch 00021: val_loss did not improve from 0.17664
Epoch 22/100
 - 10s - loss: 0.1306 - val_loss: 0.2022

Epoch 00022: val_loss did not improve from 0.17664
Epoch 23/100
 - 10s - loss: 0.1316 - val_loss: 0.2015

Epoch 00023: val_loss did not improve from 0.17664
Epoch 24/100
 - 10s - loss: 0.1304 - val_loss: 0.2275

Epoch 00024: val_loss did not improve from 0.17664
Epoch 25/100
 - 10s - loss: 0.1307 - val_loss: 0.1889

Epoch 00025: val_loss did not improve from 0.17664
Epoch 26/100
 - 10s - loss: 0.1301 - val_loss: 0.1980
```

```
Epoch 00026: val_loss did not improve from 0.17664
Epoch 27/100
 - 10s - loss: 0.1288 - val_loss: 0.1848

Epoch 00027: val_loss did not improve from 0.17664
Epoch 28/100
 - 10s - loss: 0.1300 - val_loss: 0.1861

Epoch 00028: val_loss did not improve from 0.17664
Epoch 29/100
 - 10s - loss: 0.1307 - val_loss: 0.2009

Epoch 00029: val_loss did not improve from 0.17664
Epoch 30/100
 - 10s - loss: 0.1280 - val_loss: 0.1991

Epoch 00030: val_loss did not improve from 0.17664
Epoch 31/100
 - 10s - loss: 0.1280 - val_loss: 0.2047

Epoch 00031: val_loss did not improve from 0.17664
Epoch 32/100
 - 10s - loss: 0.1302 - val_loss: 0.1833

Epoch 00032: val_loss did not improve from 0.17664
Epoch 33/100
 - 10s - loss: 0.1279 - val_loss: 0.1958

Epoch 00033: val_loss did not improve from 0.17664
Epoch 34/100
 - 10s - loss: 0.1274 - val_loss: 0.1641

Epoch 00034: val_loss improved from 0.17664 to 0.16414,
saving model to 34-0.164143860230.hdf5
Epoch 35/100
 - 10s - loss: 0.1297 - val_loss: 0.1818

Epoch 00035: val_loss did not improve from 0.16414
Epoch 36/100
 - 10s - loss: 0.1289 - val_loss: 0.1928

Epoch 00036: val_loss did not improve from 0.16414
Epoch 37/100
 - 10s - loss: 0.1277 - val_loss: 0.2068

Epoch 00037: val_loss did not improve from 0.16414
Epoch 38/100
 - 10s - loss: 0.1283 - val_loss: 0.1896

Epoch 00038: val_loss did not improve from 0.16414
Epoch 39/100
 - 10s - loss: 0.1270 - val_loss: 0.1952
```

```
Epoch 00039: val_loss did not improve from 0.16414
Epoch 40/100
 - 10s - loss: 0.1266 - val_loss: 0.2016

Epoch 00040: val_loss did not improve from 0.16414
Epoch 41/100
 - 10s - loss: 0.1274 - val_loss: 0.1749

Epoch 00041: val_loss did not improve from 0.16414
Epoch 42/100
 - 10s - loss: 0.1263 - val_loss: 0.1723

Epoch 00042: val_loss did not improve from 0.16414
Epoch 43/100
 - 10s - loss: 0.1265 - val_loss: 0.1759

Epoch 00043: val_loss did not improve from 0.16414
Epoch 44/100
 - 10s - loss: 0.1268 - val_loss: 0.1844

Epoch 00044: val_loss did not improve from 0.16414
Epoch 45/100
 - 10s - loss: 0.1282 - val_loss: 0.1597

Epoch 00045: val_loss improved from 0.16414 to 0.15974,
saving model to 45-0.159739194284.hdf5
Epoch 46/100
 - 10s - loss: 0.1277 - val_loss: 0.1662

Epoch 00046: val_loss did not improve from 0.15974
Epoch 47/100
 - 10s - loss: 0.1256 - val_loss: 0.1766

Epoch 00047: val_loss did not improve from 0.15974
Epoch 48/100
 - 10s - loss: 0.1266 - val_loss: 0.1851

Epoch 00048: val_loss did not improve from 0.15974
Epoch 49/100
 - 10s - loss: 0.1255 - val_loss: 0.1852

Epoch 00049: val_loss did not improve from 0.15974
Epoch 50/100
 - 10s - loss: 0.1269 - val_loss: 0.1895

Epoch 00050: val_loss did not improve from 0.15974
Epoch 51/100
 - 10s - loss: 0.1268 - val_loss: 0.1685

Epoch 00051: val_loss did not improve from 0.15974
Epoch 52/100
```

```
 - 10s - loss: 0.1258 - val_loss: 0.2059

Epoch 00052: val_loss did not improve from 0.15974
Epoch 53/100
 - 10s - loss: 0.1264 - val_loss: 0.1745

Epoch 00053: val_loss did not improve from 0.15974
Epoch 54/100
 - 10s - loss: 0.1275 - val_loss: 0.1541

Epoch 00054: val_loss improved from 0.15974 to 0.15406,
saving model to 54-0.154061736122.hdf5
Epoch 55/100
 - 10s - loss: 0.1261 - val_loss: 0.1741

Epoch 00055: val_loss did not improve from 0.15406
Epoch 56/100
 - 10s - loss: 0.1270 - val_loss: 0.1858

Epoch 00056: val_loss did not improve from 0.15406
Epoch 57/100
 - 10s - loss: 0.1260 - val_loss: 0.1670

Epoch 00057: val_loss did not improve from 0.15406
Epoch 58/100
 - 10s - loss: 0.1267 - val_loss: 0.1836

Epoch 00058: val_loss did not improve from 0.15406
Epoch 59/100
 - 10s - loss: 0.1281 - val_loss: 0.1600

Epoch 00059: val_loss did not improve from 0.15406
Epoch 60/100
 - 10s - loss: 0.1249 - val_loss: 0.1636

Epoch 00060: val_loss did not improve from 0.15406
Epoch 61/100
 - 10s - loss: 0.1258 - val_loss: 0.1758

Epoch 00061: val_loss did not improve from 0.15406
Epoch 62/100
 - 10s - loss: 0.1255 - val_loss: 0.1867

Epoch 00062: val_loss did not improve from 0.15406
Epoch 63/100
 - 10s - loss: 0.1255 - val_loss: 0.1721

Epoch 00063: val_loss did not improve from 0.15406
Epoch 64/100
 - 10s - loss: 0.1258 - val_loss: 0.1903

Epoch 00064: val_loss did not improve from 0.15406
```

```
Epoch 65/100
 - 10s - loss: 0.1249 - val_loss: 0.1901

Epoch 00065: val_loss did not improve from 0.15406
Epoch 66/100
 - 10s - loss: 0.1264 - val_loss: 0.1904

Epoch 00066: val_loss did not improve from 0.15406
Epoch 67/100
 - 10s - loss: 0.1261 - val_loss: 0.1569

Epoch 00067: val_loss did not improve from 0.15406
Epoch 68/100
 - 10s - loss: 0.1261 - val_loss: 0.1663

Epoch 00068: val_loss did not improve from 0.15406
Epoch 69/100
 - 10s - loss: 0.1249 - val_loss: 0.1701

Epoch 00069: val_loss did not improve from 0.15406
Epoch 70/100
 - 10s - loss: 0.1261 - val_loss: 0.1830

Epoch 00070: val_loss did not improve from 0.15406
Epoch 71/100
 - 10s - loss: 0.1253 - val_loss: 0.1857

Epoch 00071: val_loss did not improve from 0.15406
Epoch 72/100
 - 10s - loss: 0.1259 - val_loss: 0.1692

Epoch 00072: val_loss did not improve from 0.15406
Epoch 73/100
 - 10s - loss: 0.1240 - val_loss: 0.1654

Epoch 00073: val_loss did not improve from 0.15406
Epoch 74/100
 - 10s - loss: 0.1249 - val_loss: 0.2063

Epoch 00074: val_loss did not improve from 0.15406
Epoch 75/100
 - 10s - loss: 0.1245 - val_loss: 0.1846

Epoch 00075: val_loss did not improve from 0.15406
Epoch 76/100
 - 10s - loss: 0.1248 - val_loss: 0.1919

Epoch 00076: val_loss did not improve from 0.15406
Epoch 77/100
 - 10s - loss: 0.1247 - val_loss: 0.1815

Epoch 00077: val_loss did not improve from 0.15406
```

```
Epoch 78/100
 - 10s - loss: 0.1255 - val_loss: 0.1861

Epoch 00078: val_loss did not improve from 0.15406
Epoch 79/100
 - 10s - loss: 0.1254 - val_loss: 0.1843

Epoch 00079: val_loss did not improve from 0.15406
Epoch 80/100
 - 10s - loss: 0.1249 - val_loss: 0.1749

Epoch 00080: val_loss did not improve from 0.15406
Epoch 81/100
 - 10s - loss: 0.1260 - val_loss: 0.1854

Epoch 00081: val_loss did not improve from 0.15406
Epoch 82/100
 - 10s - loss: 0.1269 - val_loss: 0.2101

Epoch 00082: val_loss did not improve from 0.15406
Epoch 83/100
 - 10s - loss: 0.1252 - val_loss: 0.1990

Epoch 00083: val_loss did not improve from 0.15406
Epoch 84/100
 - 10s - loss: 0.1239 - val_loss: 0.2022

Epoch 00084: val_loss did not improve from 0.15406
Epoch 85/100
 - 10s - loss: 0.1245 - val_loss: 0.2097

Epoch 00085: val_loss did not improve from 0.15406
Epoch 86/100
 - 10s - loss: 0.1251 - val_loss: 0.2044

Epoch 00086: val_loss did not improve from 0.15406
Epoch 87/100
 - 10s - loss: 0.1244 - val_loss: 0.1768

Epoch 00087: val_loss did not improve from 0.15406
Epoch 88/100
 - 10s - loss: 0.1236 - val_loss: 0.1631

Epoch 00088: val_loss did not improve from 0.15406
Epoch 89/100
 - 10s - loss: 0.1262 - val_loss: 0.1679

Epoch 00089: val_loss did not improve from 0.15406
Epoch 90/100
 - 10s - loss: 0.1246 - val_loss: 0.2443

Epoch 00090: val_loss did not improve from 0.15406
```

```
Epoch 91/100
 - 10s - loss: 0.1265 - val_loss: 0.2047

Epoch 00091: val_loss did not improve from 0.15406
Epoch 92/100
 - 10s - loss: 0.1249 - val_loss: 0.2127

Epoch 00092: val_loss did not improve from 0.15406
Epoch 93/100
 - 10s - loss: 0.1243 - val_loss: 0.1958

Epoch 00093: val_loss did not improve from 0.15406
Epoch 94/100
 - 10s - loss: 0.1240 - val_loss: 0.1762

Epoch 00094: val_loss did not improve from 0.15406
Epoch 95/100
 - 10s - loss: 0.1250 - val_loss: 0.2092

Epoch 00095: val_loss did not improve from 0.15406
Epoch 96/100
 - 10s - loss: 0.1243 - val_loss: 0.1988

Epoch 00096: val_loss did not improve from 0.15406
Epoch 97/100
 - 10s - loss: 0.1240 - val_loss: 0.1836

Epoch 00097: val_loss did not improve from 0.15406
Epoch 98/100
 - 10s - loss: 0.1245 - val_loss: 0.1998

Epoch 00098: val_loss did not improve from 0.15406
Epoch 99/100
 - 10s - loss: 0.1245 - val_loss: 0.1929

Epoch 00099: val_loss did not improve from 0.15406
Epoch 100/100
 - 10s - loss: 0.1254 - val_loss: 0.1744

Epoch 00100: val_loss did not improve from 0.15406
Done
```

In [31]:

```
# Loading best model checkpoint

nvidia = NVIDA(lr_rate=lr, d_rate=d, verbose=0)

nvidia.load_weights("41-0.148576635343.hdf5")
scores = nvidia.evaluate(test_x,test_y,verbose=0)
print( f"{nvidia.metrics_names}: {scores}")
```

```
['loss']: 0.14857663764049028
```

```
In [32]: pred_angle = nvidia.predict(test_x)
         pred_angle[:15]
```

```
array([[-0.39942533],
       [-0.36128697],
       [-0.32552075],
       [-0.2734226 ],
       [-0.19971263],
       [-0.14588314],
       [-0.11527184],
       [-0.01421635],
       [-0.04970898],
       [ 0.06233919],
       [ 0.09712008],
       [ 0.07112201],
       [-0.16278084],
       [-0.20308086],
       [-0.21070103]], dtype=float32)
```

visualisation of predicted angle

```
In [39]: # visualisation predicted angle result
         #pip3 install opencv-python

         import scipy.misc
         import cv2
         from subprocess import call
         import math
         import imageio
         import time


         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 ra
         dius,
                 #but steering wheel angle is proportional to the invers
         e of turning radius
                 #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 = imageio.imread("driving_dataset/" + str(i) + ".jpg", pilmode="RGB")
    image = cv2.resize(full_image[-150:], dsize=(200,66)) / 255.0

    degrees = (nvidia.predict(image.reshape(1,66,200,3))[0][0]*180.0 )/scipy.pi
    cv2.imshow("frame", cv2.cvtColor(full_image, cv2.COLOR_RGB2BGR))

    #make smooth angle transitions by turning the steering wheel based on the difference of the current angle
    #and the predicted angle
    smoothed_angle += 0.2 * pow(abs((degrees - smoothed_angle)), 2.0 / 3.0) * (degrees - smoothed_angle) / abs(degrees - smoothed_angle)

    #print("Steering angle: " + str(degrees) + " (pred)\t" + str(ys[i]*180/scipy.pi) + " (actual)")
    #print("smoothed_angle: ",smoothed_angle)

    M = cv2.getRotationMatrix2D((cols/2,rows/2),-smoothed_angle,1)
    dst = cv2.warpAffine(img,M,(cols,rows))
    cv2.imshow("steering wheel", dst)
    time.sleep(0.02)
    i += 1

cv2.destroyAllWindows()
```

```
Starting frameofvideo:36325
```

Observation

1. Although traning data was not very large still Nvidia end to end model is doing quite well in prediction angle of stering wheels (ofcourse not near close to use in real life).

END :)