training

October 14, 2018

In [2]:
```
import os
import random
from numba import cuda
from PIL.Image import Image
import numpy as np
import scipy.signal as scs
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import PIL.Image as Image
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from typing import Optional, Tuple
```

In [3]:
```
%matplotlib inline
sns.set_style("whitegrid")
```

In [4]:
```
MODEL_PATH = 'models'
```

In [5]:
```
DATASET_PATH = os.path.join('.', 'dataset')
DATASET_PATH
```

Out[5]:
```
'./dataset'
```

In [6]:
```
DRIVING_LOG_FILENAME = 'driving_log.csv'
DRIVING_LOG_PATH = os.path.join(DATASET_PATH, DRIVING_LOG_FILENAME)
DRIVING_LOG_PATH
```

Out[6]:
```
'./dataset/driving_log.csv'
```

In [7]:
```
df = pd.read_csv(DRIVING_LOG_PATH, header=None,
                 names=['Center Image', 'Left Image', 'Right Image',
                        'Steering Angle',
                        'Throttle', 'Brake', 'Speed'])
df.sample(n=10)
```

Out[7]:
```
                                Center Image  \
11806  IMG/center_2018_10_10_21_05_37_378.jpg
```

```
5278   IMG/center_2018_10_07_22_30_37_447.jpg
14993  IMG/center_2018_10_12_20_41_30_938.jpg
14740  IMG/center_2018_10_12_20_40_20_200.jpg
7860   IMG/center_2018_10_07_22_39_27_458.jpg
10641  IMG/center_2018_10_10_21_04_03_398.jpg
11705  IMG/center_2018_10_10_21_05_29_532.jpg
7798   IMG/center_2018_10_07_22_39_22_510.jpg
8469   IMG/center_2018_10_07_22_40_16_120.jpg
2763   IMG/center_2018_10_07_22_22_46_715.jpg

                                   Left Image  \
11806  IMG/left_2018_10_10_21_05_37_378.jpg
5278   IMG/left_2018_10_07_22_30_37_447.jpg
14993  IMG/left_2018_10_12_20_41_30_938.jpg
14740  IMG/left_2018_10_12_20_40_20_200.jpg
7860   IMG/left_2018_10_07_22_39_27_458.jpg
10641  IMG/left_2018_10_10_21_04_03_398.jpg
11705  IMG/left_2018_10_10_21_05_29_532.jpg
7798   IMG/left_2018_10_07_22_39_22_510.jpg
8469   IMG/left_2018_10_07_22_40_16_120.jpg
2763   IMG/left_2018_10_07_22_22_46_715.jpg

                                   Right Image  Steering Angle  Throttle  Brake  \
11806  IMG/right_2018_10_10_21_05_37_378.jpg       -0.052941  1.000000    0.0
5278   IMG/right_2018_10_07_22_30_37_447.jpg       -0.288235  0.000000    0.0
14993  IMG/right_2018_10_12_20_41_30_938.jpg        0.000000  1.000000    0.0
14740  IMG/right_2018_10_12_20_40_20_200.jpg        0.000000  1.000000    0.0
7860   IMG/right_2018_10_07_22_39_27_458.jpg        0.429412  0.285795    0.0
10641  IMG/right_2018_10_10_21_04_03_398.jpg       -0.011765  1.000000    0.0
11705  IMG/right_2018_10_10_21_05_29_532.jpg       -0.035294  1.000000    0.0
7798   IMG/right_2018_10_07_22_39_22_510.jpg       -0.182353  0.000000    0.0
8469   IMG/right_2018_10_07_22_40_16_120.jpg        1.000000  0.000000    0.0
2763   IMG/right_2018_10_07_22_22_46_715.jpg        0.000000  1.000000    0.0

           Speed
11806  30.190320
5278   14.485100
14993  30.189480
14740  29.223750
7860   11.365510
10641  30.190270
11705  30.190180
7798   13.830190
8469    8.298722
2763   30.190170
```

In [8]:
```
df.dtypes
```

Out[8]:
```
Center Image       object
```

```
Left Image          object
Right Image         object
Steering Angle     float64
Throttle           float64
Brake              float64
Speed              float64
dtype: object
```

```
In [9]: df['Steering Angle'] = df['Steering Angle'].astype(np.float32)
        df['Throttle']       = df['Throttle'].astype(np.float32)
        df['Brake']          = df['Brake'].astype(np.float32)
        df['Speed']          = df['Speed'].astype(np.float32)
```

Here's an overview of the dataset. We find that we have some 15.7k training examples (for each camera position), resulting in roughly 47k training examples when all images are used. We're going to ignore throttle, brake and speed controls during training, but will still investigate them in order to see if we can harvest some information about the dataset.

```
In [10]: df.describe()
```

Out[10]:

	Steering Angle	Throttle	Brake	Speed
count	15679.000000	15679.000000	15679.000000	1.567900e+04
mean	0.013505	0.660368	0.007195	2.269208e+01
std	0.297498	0.444207	0.068826	8.494032e+00
min	-1.000000	0.000000	0.000000	3.321202e-07
25%	-0.047059	0.000000	0.000000	1.443129e+01
50%	0.000000	1.000000	0.000000	3.013469e+01
75%	0.035294	1.000000	0.000000	3.019023e+01
max	1.000000	1.000000	1.000000	3.058793e+01

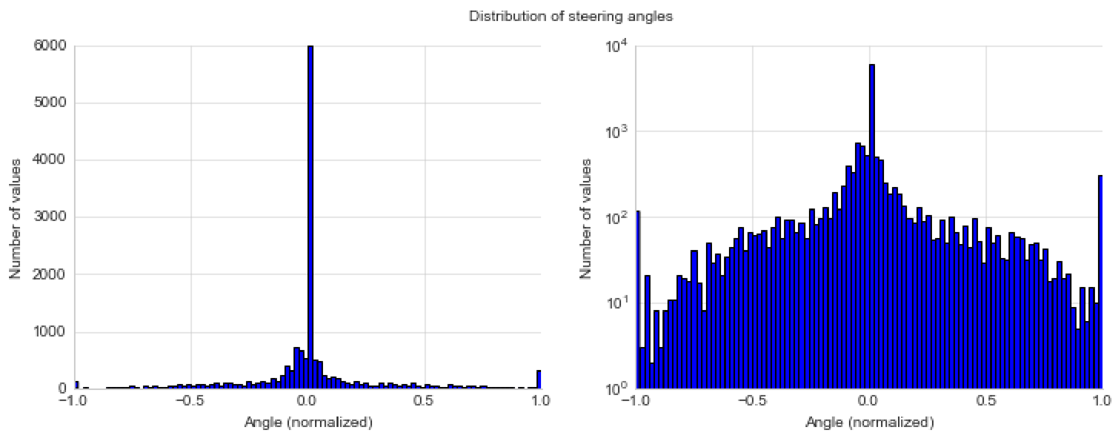
Let's see how the data is distributed. If, for example, only left turns were made (and no data augmentation is performed), a trained model would very likely be biased towards left turns.

We first define a helper function for plotting the data.

```
In [9]: def histogram(series: pd.Series, name: str, label: str, filename: Optional[str]=None, 
            f, axs = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))
            series.hist(bins=bins, ax=axs[0])
            series.hist(bins=bins, log=10, ax=axs[1])
            f.suptitle('Distribution of {}'.format(name))
            axs[0].set_ylabel('Number of values')
            axs[0].set_xlabel(label)
            axs[1].set_ylabel('Number of values')
            axs[1].set_xlabel(label)
            sns.despine()
            if filename is not None:
                plt.savefig(os.path.join('examples', filename))
```

When we look at the steering angles, we quickly find that most of the steering angles appear to be exactly or very close to zero. The distribution of angles also seems to be slightly skewed to the left, which we will mitigate by mirroring the images during training. We can also see some minor spikes at both ends of the spectrum indicating that extreme angles were used much more often than moderate ones.

```
In [10]: histogram(df['Steering Angle'], 'steering angles', 'Angle (normalized)', 'steering-ang
```



Here's the number of occurrences and ranges numerically when using 100 bins as above:

```
In [11]: counts, ranges = np.histogram(df['Steering Angle'], bins=100)
         counts, ranges

Out[11]: (array([ 116,    3,   21,    2,    8,    3,    8,   11,   11,   21,   19,
                   18,   41,   17,    8,   50,   29,   37,   21,   34,   45,   57,
                   75,   40,   67,   61,   63,   69,   44,   76,  100,   57,   93,
                   91,   68,   87,   57,  126,   83,   97,  131,   95,  194,  125,
                  234,  398,  329,  719,  673,  534, 5991,  496,  469,  250,  187,
                  222,  184,  133,   98,   87,  128,   88,  103,   54,   57,   91,
                   49,  102,   68,   47,   78,   44,   96,   52,   29,   74,   50,
                   61,   33,   32,   66,   58,   57,   32,   47,   51,   32,   43,
                   18,   19,   31,   19,   22,    9,    5,   15,    6,   15,   10,
                  305]),
          array([-1.  , -0.98, -0.96, -0.94, -0.92, -0.9 , -0.88, -0.86, -0.84,
                 -0.82, -0.8 , -0.78, -0.76, -0.74, -0.72, -0.7 , -0.68, -0.66,
                 -0.64, -0.62, -0.6 , -0.58, -0.56, -0.54, -0.52, -0.5 , -0.48,
                 -0.46, -0.44, -0.42, -0.4 , -0.38, -0.36, -0.34, -0.32, -0.3 ,
                 -0.28, -0.26, -0.24, -0.22, -0.2 , -0.18, -0.16, -0.14, -0.12,
                 -0.1 , -0.08, -0.06, -0.04, -0.02,  0.  ,  0.02,  0.04,  0.06,
                  0.08,  0.1 ,  0.12,  0.14,  0.16,  0.18,  0.2 ,  0.22,  0.24,
                  0.26,  0.28,  0.3 ,  0.32,  0.34,  0.36,  0.38,  0.4 ,  0.42,
                  0.44,  0.46,  0.48,  0.5 ,  0.52,  0.54,  0.56,  0.58,  0.6 ,
                  0.62,  0.64,  0.66,  0.68,  0.7 ,  0.72,  0.74,  0.76,  0.78,
                  0.8 ,  0.82,  0.84,  0.86,  0.88,  0.9 ,  0.92,  0.94,  0.96,
                  0.98,  1.  ], dtype=float32))
```

When we median filter the occurrences, we find that the peak gets replaced with a value of 494.

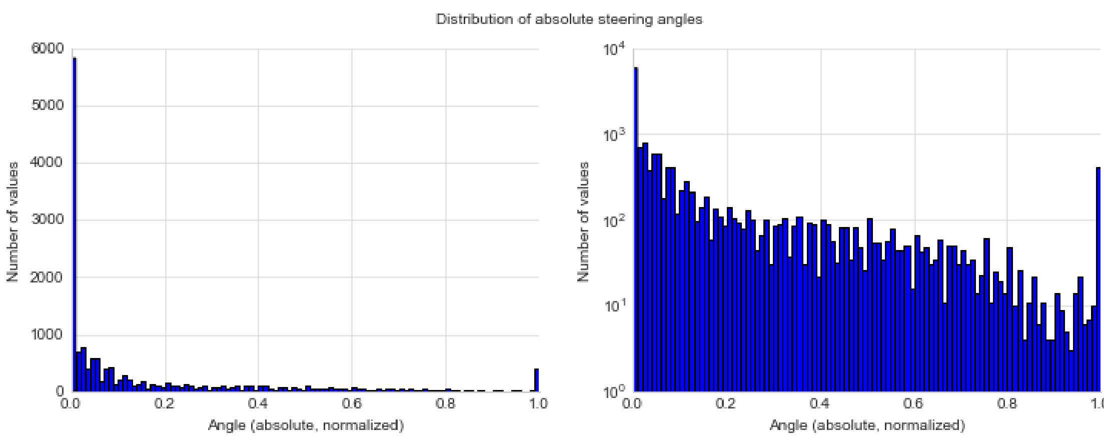
```
In [12]: scs.medfilt(counts)

Out[12]: array([  3.,   21.,    3.,    8.,    3.,    8.,    8.,   11.,   11.,   19.,   19.,
                 19.,   18.,   17.,   17.,   29.,   37.,   29.,   34.,   34.,   45.,   57.,
                 57.,   67.,   61.,   63.,   63.,   63.,   69.,   76.,   76.,   93.,   91.,
                 91.,   87.,   68.,   87.,   83.,   97.,   97.,   97.,  131.,  125.,  194.,
                234.,  329.,  398.,  673.,  673.,  673.,  534.,  496.,  469.,  250.,  222.,
                187.,  184.,  133.,   98.,   98.,   88.,  103.,   88.,   57.,   57.,   57.,
                 91.,   68.,   68.,   68.,   47.,   78.,   52.,   52.,   52.,   50.,   61.,
                 50.,   33.,   33.,   58.,   58.,   57.,   47.,   47.,   47.,   43.,   32.,
                 19.,   19.,   19.,   22.,   19.,    9.,    9.,    6.,   15.,   10.,   15.,
                 10.])
```

Given that this is about 10% of the original bin size, we'll be stochastically undersampling the "zero" angles by 90% (i.e. keep only randomly chosen 10% of the data) in order to prevent bias.

```
In [13]: 494 / 5205

Out[13]: 0.09490874159462055
```

When we look at the distribution of the absolute angles we find that the amounts appear to be log-linearly decreasing with increasing absolute angle, except for the peak at maximum angle. We could oversample angles with few occurrences, but given that these should be used only in extreme situations anyways, it might make sense to keep the distribution like that.

```
In [14]: histogram(df['Steering Angle'].abs(), 'absolute steering angles', 'Angle (absolute, no
```



When we look at the statistics of the absolute steering angles, we find that only 50% of the values are greater than about 0.04. Since the simulator steering angles range from -25 to 25 (degrees, presumable), this would correspond to an angle of 0.88235 degrees (again, presumably).

```
In [15]: df['Steering Angle'].abs().describe()

Out[15]: count    15679.000000
         mean         0.162649
         std          0.249462
         min          0.000000
         25%          0.000000
         50%          0.041176
         75%          0.217647
         max          1.000000
         Name: Steering Angle, dtype: float64
```

We see that for the extreme "angle" of +1 we have more values, resulting in a mean value that is slightly positive, despite the "bump" on the negative side of the distribution.

```
In [16]: df['Steering Angle'].describe()

Out[16]: count    15679.000000
         mean         0.013505
         std          0.297498
         min         -1.000000
         25%         -0.047059
         50%          0.000000
         75%          0.035294
         max          1.000000
         Name: Steering Angle, dtype: float64
```

As stated already, mirroring the image data during training will duplicate the number of examples and iron out these small issues. Even without that, the dataset appears to be balanced; care needs to be taken not to oversample the "zero" angles. As they make up for more than one third (about 36%) of the training data, this could introduce a severe bias.

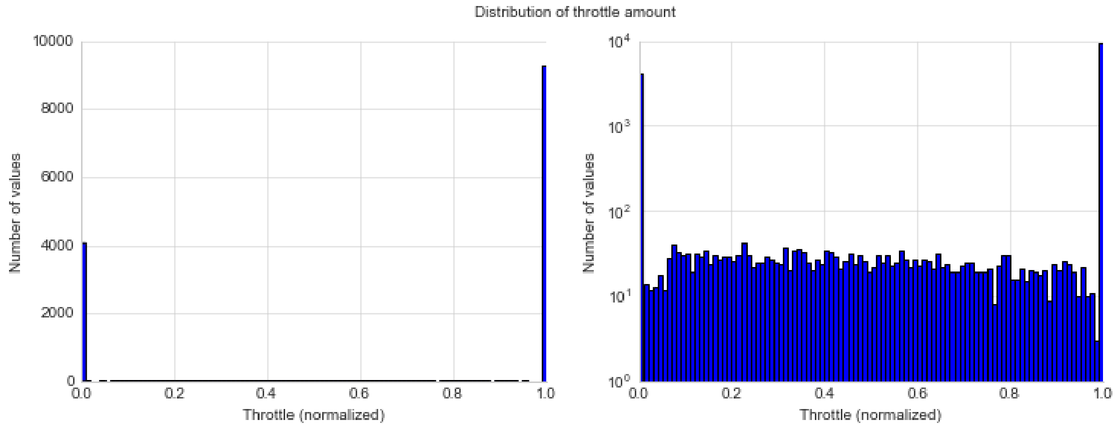
```
In [17]: sum(df['Steering Angle'] == 0) / len(df['Steering Angle'])

Out[17]: 0.3530199630078449
```

For generating steering angle noise, we can later use the human driver's steering angle standard deviation to generate a gaussian to sample noise from.

```
In [18]: steering_std = df['Steering Angle'].std()
         steering_std

Out[18]: 0.29749835
```

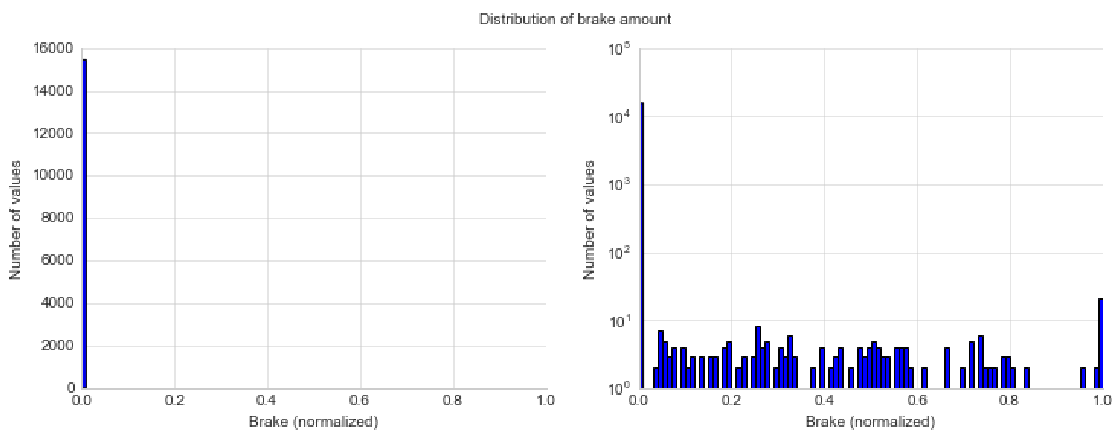
The throttle values show an entirely different picture: The values are either exactly zero or exactly one, indicating that the car was mostly standing still or driving at full speed during the training data collection. When looking at the log-transformed data, we see that there are indeed some values in between, but they are most likely sampled only during acceleration phases. We might want to undersample examples of zero throttle as we don't want the model to learn how to *not* move.

```
In [19]: histogram(df['Throttle'], 'throttle amount', 'Throttle (normalized)', 'throttle-values
```



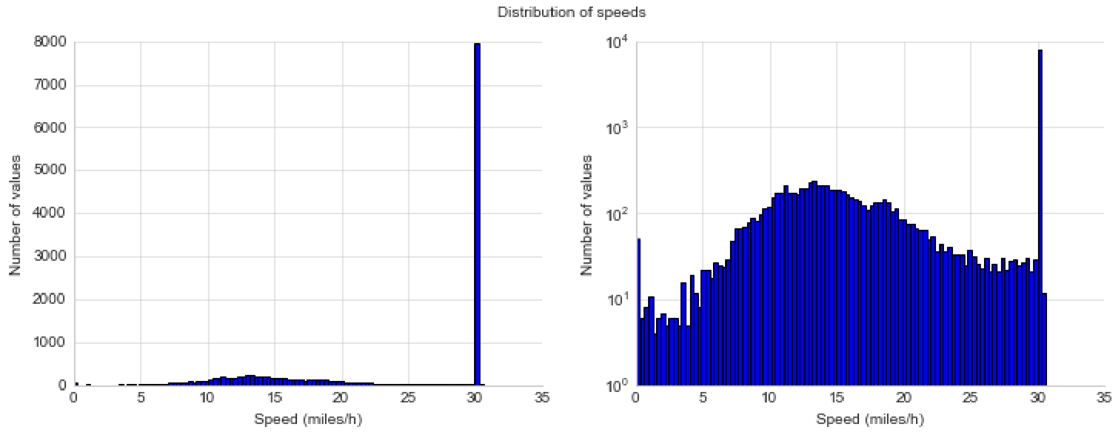
The brake values appear are mostly zero. This makes sense: We already know the car was moving mostly at full speed, so the brake was only used for a fraction of the time.

```
In [20]: histogram(df['Brake'], 'brake amount', 'Brake (normalized)', 'brake-values.png')
```



Interestingly for the speeds we find a big amount of data distributed around 15 miles/hour. This might be due to data collection on the optional track: As that one is much harder to maneuver through, lower speeds were used there. The high spike for full speed corresponds to the high spike for full throttle. Zero speed situations are of no use to us.

```
In [21]: histogram(df['Speed'], 'speeds', 'Speed (miles/h)', 'speed-values.png')
```



0.1 Sampling training data

We're going to sample data based on zero or nonzero steering angle.

```
In [22]: zero_angles = (df['Steering Angle'] == 0)
         sum(zero_angles)

Out[22]: 5535
```

As stated before, we're generally only interested in rows with a positive speed.

```
In [23]: positive_speed = (df['Speed'] > 0)
         sum(positive_speed)

Out[23]: 15679
```

We now split the dataset into two groups: Zero and nonzero steering angles.

```
In [24]: sample_seed = 0
         df_nonzero_angles = df[positive_speed & ~zero_angles]
         df_zero_angles    = df[positive_speed & zero_angles]
```

We're going to split both sets into two groups once more: Training and validation data.

```
In [25]: df_train_nonzero, df_valid_nonzero = train_test_split(df_nonzero_angles,
                                                  test_size=0.3, random_state=samp
         df_train_zero, df_valid_zero       = train_test_split(df_zero_angles,
                                                  test_size=0.3, random_state=samp
```

We now split the validation set again into the actual validation set (used during training) and a test set (used to validate the final model).

```
In [26]: df_test_nonzero, df_valid_nonzero = train_test_split(df_valid_nonzero,
                                                               test_size=0.3, random_state=sampl
         df_test_zero, df_valid_zero       = train_test_split(df_valid_zero,
                                                               test_size=0.3, random_state=sampl
```

The “zero” training dataset still has too many elements. We’re going to take a random sample of it for training.

```
In [27]: zero_speed_factor = 0.1
         zero_speed_count_train = int(zero_speed_factor * len(df_train_zero))
         zero_speed_count_valid = int(zero_speed_factor * len(df_valid_zero))
         zero_speed_count_test  = int(zero_speed_factor * len(df_test_zero))
```

```
In [28]: df_train_zero = df_train_zero.sample(zero_speed_count_train, random_state=sample_seed)
         df_valid_zero = df_valid_zero.sample(zero_speed_count_valid, random_state=sample_seed)
         df_test_zero  = df_test_zero.sample(zero_speed_count_test, random_state=sample_seed)
```

Here’s the actual size of the “nonzero” training, validation and test sets now:

```
In [29]: len(df_train_nonzero), len(df_valid_nonzero), len(df_test_nonzero)

Out[29]: (7100, 914, 2130)
```

These are the counts of the “zero” sets:

```
In [30]: len(df_train_zero), len(df_valid_zero), len(df_test_zero)

Out[30]: (387, 49, 116)
```

We now combine the data into the actual data sets used for training.

```
In [31]: df_train = pd.concat([df_train_zero, df_train_nonzero])
         df_valid = pd.concat([df_valid_zero, df_valid_nonzero])
         df_test = pd.concat([df_test_zero, df_test_nonzero])
         len(df_train), len(df_valid), len(df_test)

Out[31]: (7487, 963, 2246)
```

For the first iteration of training, we’re only going to train with images from the center camera.

```
In [32]: import keras

Using TensorFlow backend.
```

We’ll be using a custom data generator as described in A detailed example of how to use data generators with Keras.

```
In [33]: class DataGenerator(keras.utils.Sequence):
             'Generates data for Keras'
             def __init__(self, df: pd.DataFrame, batch_size: int=32, dim: Tuple[int, int]=(32
                          dataset_path: str=DATASET_PATH):
                 self.batch_size = batch_size
                 self.df = df
                 self.dim = dim
                 self.shuffle = shuffle
                 self.dataset_path = dataset_path
                 self.on_epoch_end()

             def __len__(self):
                 """Denotes the number of batches per epoch"""
                 return int(np.floor(len(self.df) / self.batch_size))

             def __getitem__(self, index):
                 """Generate one batch of data"""
                 rows = self.df.iloc[index*self.batch_size:(index+1)*self.batch_size]
                 return self.__data_generation(rows)

             def on_epoch_end(self):
                 """Updates indexes after each epoch"""
                 if self.shuffle == True:
                     # self.df = self.df.sample(frac=1).reset_index(drop=True)
                     self.df = shuffle(self.df)

             def __data_generation(self, rows) -> Tuple[np.ndarray, np.float32]:
                 """Generates data containing batch_size samples"""
                 # Initialization
                 width, height, channels = self.dim
                 X = np.empty((self.batch_size, height, width, channels), dtype=np.uint8)
                 y = np.empty((self.batch_size), dtype=np.float32)

                 # Generate data
                 for i, (idx, row) in enumerate(rows.iterrows()):
                     # Store sample and angle
                     X[i,], y[i] = self._get_example(row)

                 return X, y

             def _get_example(self, row) -> Tuple[np.ndarray, np.float32]:
                 path = os.path.join(self.dataset_path, row['Center Image'])
                 img = Image.open(path).convert('RGB')
                 X = np.array(img).astype(np.uint8)
                 y = row['Steering Angle']
                 return X, y
```

Let's test the generator:

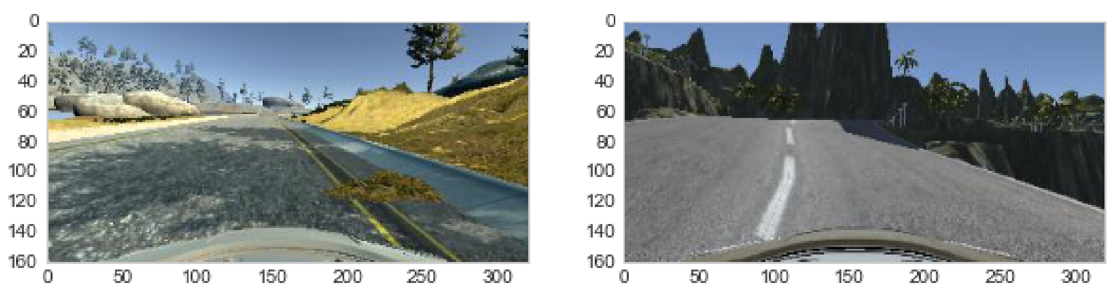
```
In [34]: test_generator = DataGenerator(df_train, batch_size=2)
         print('Number of samples:', len(test_generator))
```

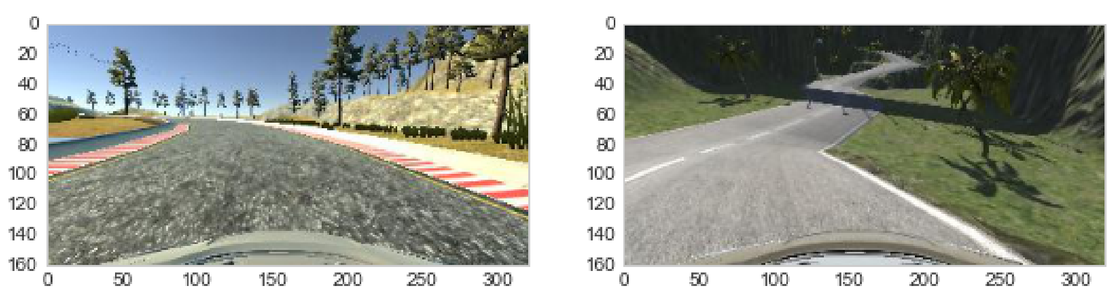
```
Number of samples: 3743
```

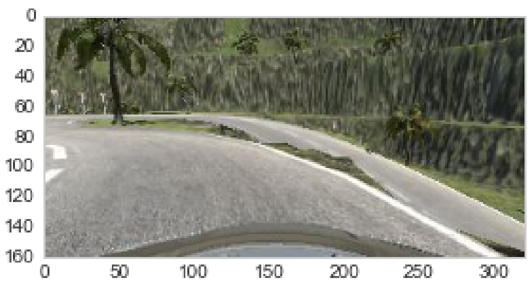
Let's enumerate some of the examples:

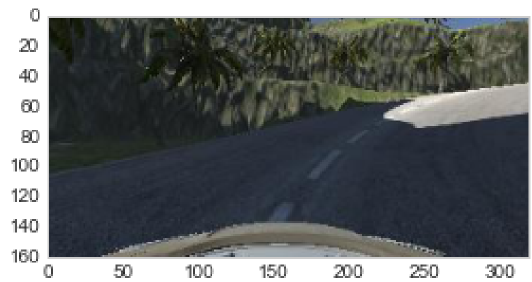
```
In [35]: i = 0
         for i, (X, y) in enumerate(test_generator):
             if i == 0:
                 print(X.shape, y.shape)
             f, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))
             axs[0].imshow(X[0])
             axs[1].imshow(X[1])
             axs[0].grid(None)
             axs[1].grid(None)
             i += 1
             if i == 3:
                 break
```

```
(2, 160, 320, 3) (2,)
```









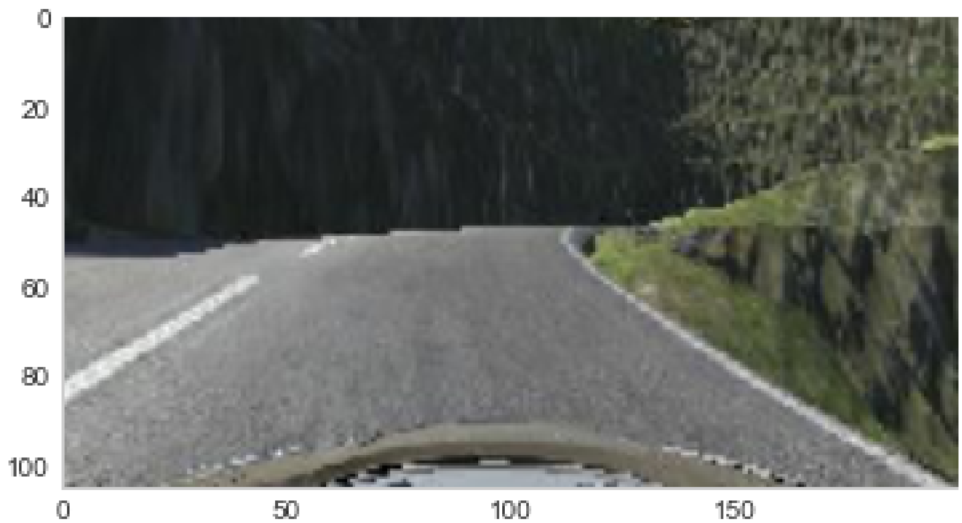
Since the sky region is not containing any information relevant to steering, we can remove it. A possible exception to this assumption would be situations where the car is driving down a hill; it might be worth exploring that later. Likewise, the bottom part of the image always contains the hood of the car and is thus not relevant to steering. If we decide to use images from the left and right camera as well, this region might be problematic as well, as the network could learn to be biased towards the position of the hood.

In [36]:
```
NVIDIA_WIDTH  = 200
NVIDIA_HEIGHT = 66
NVIDIA_ASPECT = NVIDIA_HEIGHT / NVIDIA_WIDTH
```

In [37]:
```
NEW_WIDTH = int(NVIDIA_WIDTH)
NEW_HEIGHT = int(320 * NVIDIA_ASPECT)
print('{}, {}'.format(NEW_WIDTH, NEW_HEIGHT))
```

```
200, 105
```

In [38]:
```
test_img_path = os.path.join(DATASET_PATH, df_train.iloc[1]['Center Image'])
test_img = Image.open(test_img_path).convert('RGB').resize((NEW_WIDTH, NEW_HEIGHT))
test_img = np.array(test_img).astype(np.float32) / 255.
plt.imshow(test_img)
plt.grid(None)
sns.despine()
```

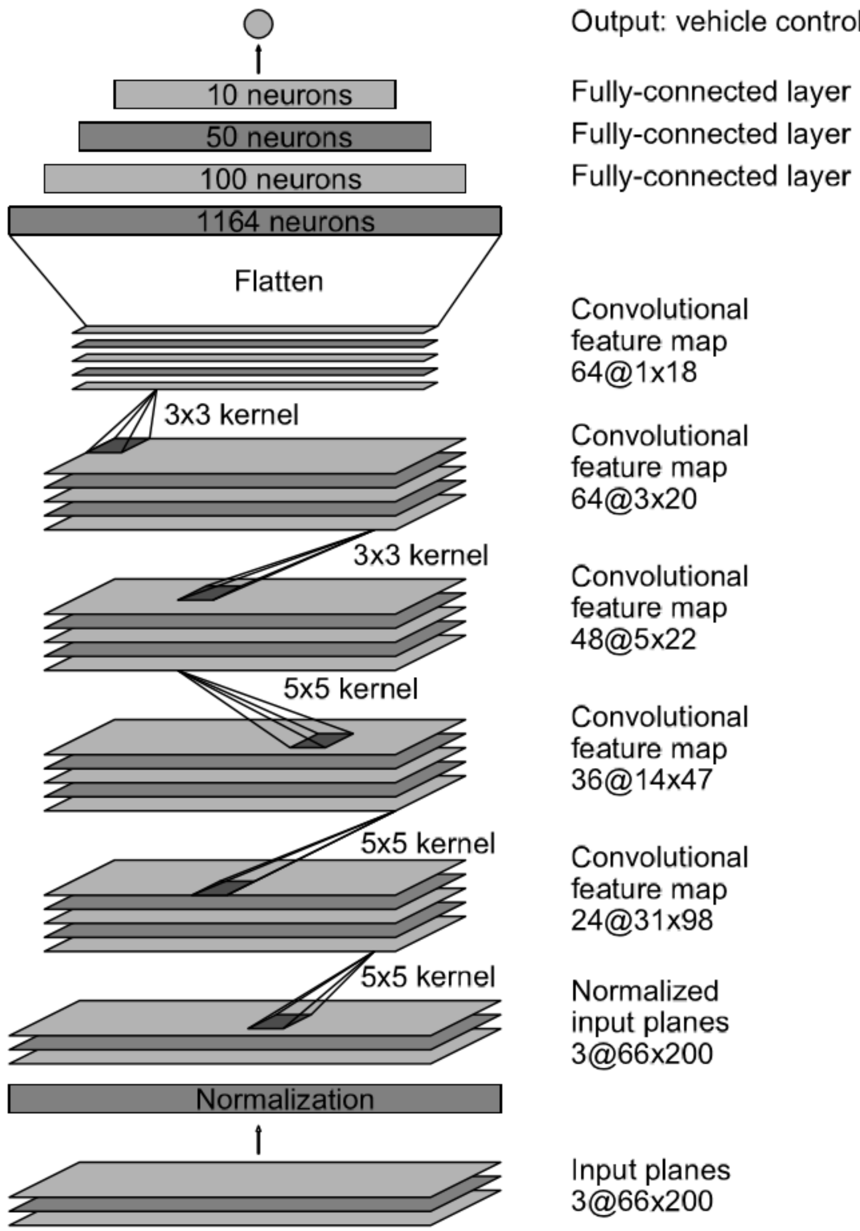



```
In [39]: CROP_TOP = 24
         CROP_BOTTOM = 15

         cropped_test_img = test_img[CROP_TOP:-CROP_BOTTOM, ...]
         print('Cropped image size: {}'.format(cropped_test_img.shape))
         plt.imshow(cropped_test_img)
         plt.grid(None)
         sns.despine()
```

```
Cropped image size: (66, 200, 3)
```

0.2 Building a baseline network architecture

As a baseline, we're going to implement the model suggested by the End to End Learning for Self-Driving Cars published by NVIDIA:



Network architecture for “End to End Learning for Self-Driving Cars”

In [40]:
```python
BATCH_SIZE = 128
EPOCHS = 100
LEARNING_RATE = 1e-3
```

In [41]:
```python
import tensorflow as tf
from keras import backend as K
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Flatten, Dense, Lambda, Activation, Conv2D, Cropping2D
```

The paper specifies image input is converted to YUV prior to processing.

In [42]:
```python
def yuv_conversion(x):
    import tensorflow as tf
    x = tf.cast(x, dtype=tf.float32) / 255.0
    return tf.image.rgb_to_yuv(x)
```

In order to be as close to the original paper as possible, we'll be resizing the image to 200 pixels in width such that after cropping, the input image size will 200x66.

In [43]:
```python
def image_resize(x):
    return K.tf.image.resize_images(x, (105, 200))
```

In [51]:
```python
def nvidia_model(learning_rate: float, decay: float=0):
    model = Sequential()

    # Resizing the images to the NVIDIA proposed shape:
    model.add(Lambda(image_resize, input_shape = (160, 320, 3), name='resize_image'))

    # YUV conversion and normalization
    model.add(Lambda(yuv_conversion, input_shape = (105, 200, 3), name='rgb_to_yuv'))
    model.add(Lambda(lambda x: x * 2 - 1, input_shape = (105, 200, 3), name='normalize

    # Crop the image to remove the sky and car hood
    model.add(Cropping2D(cropping=((CROP_TOP, CROP_BOTTOM), (0, 0)),
                         input_shape=(105, 200, 3), name='crop'))

    model.add(Conv2D(24, (5, 5), strides=(2, 2), padding='valid', name='conv_1'))
    model.add(Activation('relu', name='conv_1_relu'))

    model.add(Conv2D(36, (5, 5), strides=(2, 2), padding='valid', name='conv_2'))
    model.add(Activation('relu', name='conv_2_relu'))

    model.add(Conv2D(48, (5, 5), strides=(2, 2), padding='valid', name='conv_3'))
    model.add(Activation('relu', name='conv_3_relu'))

    model.add(Conv2D(64, (3, 3), strides=(1, 1), padding='valid', name='conv_4'))
    model.add(Activation('relu', name='conv_4_relu'))
```

```
        model.add(Conv2D(64, (3, 3), strides=(1, 1), padding='valid', name='conv_5'))
        model.add(Activation('relu', name='conv_5_relu'))

        model.add(Flatten())

        model.add(Dense(100, name='fc_1'))
        model.add(Activation('relu', name='fc_1_relu'))

        model.add(Dense(50, name='fc_2'))
        model.add(Activation('relu', name='fc_2_relu'))

        model.add(Dense(10, name='fc_3'))
        model.add(Activation('relu', name='fc_3_relu'))

        model.add(Dense(1, name='angle'))

        adam = Adam(lr=learning_rate, decay=decay)
        model.compile(optimizer=adam, loss='mse')

        return model

    K.clear_session()
    model = nvidia_model(LEARNING_RATE)
    model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resize_image (Lambda)        (None, 105, 200, 3)       0
_________________________________________________________________
rgb_to_yuv (Lambda)          (None, 105, 200, 3)       0
_________________________________________________________________
normalize (Lambda)           (None, 105, 200, 3)       0
_________________________________________________________________
crop (Cropping2D)            (None, 66, 200, 3)        0
_________________________________________________________________
conv_1 (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv_1_relu (Activation)     (None, 31, 98, 24)        0
_________________________________________________________________
conv_2 (Conv2D)              (None, 14, 47, 36)        21636
_________________________________________________________________
conv_2_relu (Activation)     (None, 14, 47, 36)        0
_________________________________________________________________
conv_3 (Conv2D)              (None, 5, 22, 48)         43248
_________________________________________________________________
conv_3_relu (Activation)     (None, 5, 22, 48)         0
```

```
_________________________________________________________________
conv_4 (Conv2D)              (None, 3, 20, 64)         27712
_________________________________________________________________
conv_4_relu (Activation)     (None, 3, 20, 64)         0
_________________________________________________________________
conv_5 (Conv2D)              (None, 1, 18, 64)         36928
_________________________________________________________________
conv_5_relu (Activation)     (None, 1, 18, 64)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
fc_1 (Dense)                 (None, 100)               115300
_________________________________________________________________
fc_1_relu (Activation)       (None, 100)               0
_________________________________________________________________
fc_2 (Dense)                 (None, 50)                5050
_________________________________________________________________
fc_2_relu (Activation)       (None, 50)                0
_________________________________________________________________
fc_3 (Dense)                 (None, 10)                510
_________________________________________________________________
fc_3_relu (Activation)       (None, 10)                0
_________________________________________________________________
angle (Dense)                (None, 1)                 11
=================================================================
Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0
_________________________________________________________________
```

In this setup, the number of parameters after flattening (`1152`) differs from the NVIDIA paper (`1164`). However, since the paper explicitly states a convolution output of `64@1x18` was obtained, the number reported by NVIDIA appears to be an error.

We can now create the training and validation data generators.

```
In [45]: training_generator   = DataGenerator(df_train, batch_size=BATCH_SIZE)
         validation_generator = DataGenerator(df_valid, batch_size=BATCH_SIZE)
```

We'll be using a `ModelCheckpoint` to save a model whenever the validation loss decreases.

An issue exists for Keras 2.2.4 that prevents writing checkpoint files when `ModelCheckpoint` is used with `fit_generator()` and `use_multiprocessing=True` (see here). One suggested workaround is to use formatted file names, which is what we'll do.

```
In [46]: CHECKPOINT_PATH = os.path.join(MODEL_PATH, 'nvidia.{epoch:02d}-{val_loss:.4f}.h5')
         checkpoint = ModelCheckpoint(CHECKPOINT_PATH, monitor = 'val_loss', verbose=1,
                                      save_best_only=True, mode='min')
```

In addition, early stopping will be used to terminate training if the validation loss doesn't improve for multiple epochs.

```
In [47]: early_stopping = EarlyStopping(monitor='val_loss', patience=20, verbose=1,
                                        mode='min', restore_best_weights=False)
```

We can now run the training.

```
In [48]: hist = model.fit_generator(generator=training_generator,
                                   validation_data=validation_generator,
                                   use_multiprocessing=True, workers=6,
                                   callbacks=[checkpoint, early_stopping],
                                   epochs=EPOCHS, verbose=1)
```

```
Epoch 1/100
58/58 [==============================] - 27s 470ms/step - loss: 0.0888 - val_loss: 0.0610

Epoch 00001: val_loss improved from inf to 0.06097, saving model to models/nvidia.01-0.0610.h5
Epoch 2/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0563 - val_loss: 0.0527

Epoch 00002: val_loss improved from 0.06097 to 0.05267, saving model to models/nvidia.02-0.0527
Epoch 3/100
58/58 [==============================] - 21s 358ms/step - loss: 0.0489 - val_loss: 0.0437

Epoch 00003: val_loss improved from 0.05267 to 0.04374, saving model to models/nvidia.03-0.0437
Epoch 4/100
58/58 [==============================] - 21s 361ms/step - loss: 0.0430 - val_loss: 0.0415

Epoch 00004: val_loss improved from 0.04374 to 0.04148, saving model to models/nvidia.04-0.0415
Epoch 5/100
58/58 [==============================] - 22s 377ms/step - loss: 0.0364 - val_loss: 0.0431

Epoch 00005: val_loss did not improve from 0.04148
Epoch 6/100
57/58 [============================>.] - ETA: 0s - loss: 0.0353
Epoch 6/100
58/58 [==============================] - 22s 378ms/step - loss: 0.0352 - val_loss: 0.0419

Epoch 00006: val_loss did not improve from 0.04148
Epoch 7/100
58/58 [==============================] - 19s 334ms/step - loss: 0.0300 - val_loss: 0.0369

Epoch 00007: val_loss improved from 0.04148 to 0.03689, saving model to models/nvidia.07-0.0369
Epoch 8/100
58/58 [==============================] - 22s 374ms/step - loss: 0.0274 - val_loss: 0.0351

Epoch 00008: val_loss improved from 0.03689 to 0.03510, saving model to models/nvidia.08-0.0351
Epoch 9/100
58/58 [==============================] - 21s 361ms/step - loss: 0.0263 - val_loss: 0.0353
```

```
Epoch 00009: val_loss did not improve from 0.03510
Epoch 10/100
58/58 [==============================] - 21s 360ms/step - loss: 0.0232 - val_loss: 0.0332

Epoch 00010: val_loss improved from 0.03510 to 0.03319, saving model to models/nvidia.10-0.033
Epoch 11/100
58/58 [==============================] - 21s 364ms/step - loss: 0.0212 - val_loss: 0.0315

Epoch 00011: val_loss improved from 0.03319 to 0.03149, saving model to models/nvidia.11-0.031
Epoch 12/100
58/58 [==============================] - 22s 372ms/step - loss: 0.0202 - val_loss: 0.0297

Epoch 00012: val_loss improved from 0.03149 to 0.02975, saving model to models/nvidia.12-0.029
Epoch 13/100
58/58 [==============================] - 21s 370ms/step - loss: 0.0179 - val_loss: 0.0347

Epoch 00013: val_loss did not improve from 0.02975
Epoch 14/100
58/58 [==============================] - 19s 336ms/step - loss: 0.0173 - val_loss: 0.0302

Epoch 00014: val_loss did not improve from 0.02975
Epoch 15/100
57/58 [============================>.] - ETA: 0s - loss: 0.0154
Epoch 00014: val_loss did not improve from 0.02975
Epoch 15/100
58/58 [==============================] - 21s 357ms/step - loss: 0.0154 - val_loss: 0.0294

Epoch 00015: val_loss improved from 0.02975 to 0.02938, saving model to models/nvidia.15-0.029
Epoch 16/100
58/58 [==============================] - 21s 365ms/step - loss: 0.0143 - val_loss: 0.0310

Epoch 00016: val_loss did not improve from 0.02938
Epoch 17/100
58/58 [==============================] - 21s 367ms/step - loss: 0.0128 - val_loss: 0.0285

Epoch 00017: val_loss improved from 0.02938 to 0.02854, saving model to models/nvidia.17-0.028
Epoch 18/100
58/58 [==============================] - 20s 344ms/step - loss: 0.0115 - val_loss: 0.0263

Epoch 00018: val_loss improved from 0.02854 to 0.02635, saving model to models/nvidia.18-0.026
Epoch 19/100
58/58 [==============================] - 20s 345ms/step - loss: 0.0107 - val_loss: 0.0285

Epoch 00019: val_loss did not improve from 0.02635
Epoch 20/100
58/58 [==============================] - 20s 349ms/step - loss: 0.0101 - val_loss: 0.0285

Epoch 00019: val_loss did not improve from 0.02635
```

```
Epoch 20/100

Epoch 00020: val_loss did not improve from 0.02635
Epoch 21/100
58/58 [==============================] - 21s 354ms/step - loss: 0.0084 - val_loss: 0.0288

Epoch 00021: val_loss did not improve from 0.02635
Epoch 22/100
58/58 [==============================] - 20s 349ms/step - loss: 0.0079 - val_loss: 0.0246

Epoch 00022: val_loss improved from 0.02635 to 0.02463, saving model to models/nvidia.22-0.024
Epoch 23/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0076 - val_loss: 0.0284

Epoch 00023: val_loss did not improve from 0.02463
Epoch 24/100
58/58 [==============================] - 20s 343ms/step - loss: 0.0073 - val_loss: 0.0269

Epoch 00024: val_loss did not improve from 0.02463
Epoch 25/100
58/58 [==============================] - 21s 354ms/step - loss: 0.0062 - val_loss: 0.0306

Epoch 00025: val_loss did not improve from 0.02463
Epoch 26/100
58/58 [==============================] - 20s 350ms/step - loss: 0.0065 - val_loss: 0.0253

Epoch 00026: val_loss did not improve from 0.02463
Epoch 27/100
58/58 [==============================] - 20s 350ms/step - loss: 0.0058 - val_loss: 0.0274

Epoch 00027: val_loss did not improve from 0.02463
Epoch 28/100
58/58 [==============================] - 20s 347ms/step - loss: 0.0050 - val_loss: 0.0249

Epoch 00028: val_loss did not improve from 0.02463
Epoch 29/100
58/58 [==============================] - 20s 353ms/step - loss: 0.0044 - val_loss: 0.0256

Epoch 00029: val_loss did not improve from 0.02463
Epoch 30/100
58/58 [==============================] - 20s 344ms/step - loss: 0.0049 - val_loss: 0.0240

Epoch 00030: val_loss improved from 0.02463 to 0.02398, saving model to models/nvidia.30-0.024
Epoch 31/100
57/58 [============================>.] - ETA: 0s - loss: 0.0046
Epoch 00030: val_loss improved from 0.02463 to 0.02398, saving model to models/nvidia.30-0.024
58/58 [==============================] - 20s 340ms/step - loss: 0.0046 - val_loss: 0.0259
```

```
Epoch 00031: val_loss did not improve from 0.02398
Epoch 32/100
57/58 [============================>.] - ETA: 0s - loss: 0.0041
Epoch 00031: val_loss did not improve from 0.02398
Epoch 32/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0042 - val_loss: 0.0239

Epoch 00032: val_loss improved from 0.02398 to 0.02392, saving model to models/nvidia.32-0.0239
Epoch 33/100
58/58 [==============================] - 20s 347ms/step - loss: 0.0039 - val_loss: 0.0245

Epoch 00033: val_loss did not improve from 0.02392
Epoch 34/100
57/58 [============================>.] - ETA: 0s - loss: 0.0037
Epoch 00033: val_loss did not improve from 0.02392
Epoch 34/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0037 - val_loss: 0.0236

Epoch 00034: val_loss improved from 0.02392 to 0.02357, saving model to models/nvidia.34-0.0236
Epoch 35/100
57/58 [============================>.] - ETA: 0s - loss: 0.0034
Epoch 00034: val_loss improved from 0.02392 to 0.02357, saving model to models/nvidia.34-0.0236
58/58 [==============================] - 20s 347ms/step - loss: 0.0035 - val_loss: 0.0242

Epoch 00035: val_loss did not improve from 0.02357
Epoch 36/100
58/58 [==============================] - 21s 354ms/step - loss: 0.0029 - val_loss: 0.0235

Epoch 00036: val_loss improved from 0.02357 to 0.02353, saving model to models/nvidia.36-0.0235
Epoch 37/100
58/58 [==============================] - 20s 347ms/step - loss: 0.0026 - val_loss: 0.0255

Epoch 00037: val_loss did not improve from 0.02353
Epoch 38/100
58/58 [==============================] - 22s 377ms/step - loss: 0.0028 - val_loss: 0.0262

Epoch 00038: val_loss did not improve from 0.02353
Epoch 39/100
58/58 [==============================] - 21s 369ms/step - loss: 0.0033 - val_loss: 0.0246

Epoch 00039: val_loss did not improve from 0.02353
Epoch 40/100
58/58 [==============================] - 21s 360ms/step - loss: 0.0036 - val_loss: 0.0244

Epoch 00040: val_loss did not improve from 0.02353
Epoch 41/100
58/58 [==============================] - 20s 353ms/step - loss: 0.0035 - val_loss: 0.0245
```

```
Epoch 00041: val_loss did not improve from 0.02353
Epoch 42/100
58/58 [==============================] - 20s 350ms/step - loss: 0.0028 - val_loss: 0.0249

Epoch 00042: val_loss did not improve from 0.02353
Epoch 43/100
58/58 [==============================] - 20s 338ms/step - loss: 0.0025 - val_loss: 0.0237

Epoch 00043: val_loss did not improve from 0.02353
Epoch 44/100
58/58 [==============================] - 20s 344ms/step - loss: 0.0023 - val_loss: 0.0228

Epoch 00044: val_loss improved from 0.02353 to 0.02279, saving model to models/nvidia.44-0.0228
Epoch 45/100
58/58 [==============================] - 20s 339ms/step - loss: 0.0023 - val_loss: 0.0247

Epoch 00045: val_loss did not improve from 0.02279
Epoch 46/100
58/58 [==============================] - 20s 342ms/step - loss: 0.0025 - val_loss: 0.0236

Epoch 00046: val_loss did not improve from 0.02279
Epoch 47/100
58/58 [==============================] - 20s 348ms/step - loss: 0.0020 - val_loss: 0.0231

Epoch 00047: val_loss did not improve from 0.02279
Epoch 48/100
58/58 [==============================] - 20s 344ms/step - loss: 0.0020 - val_loss: 0.0221

Epoch 00048: val_loss improved from 0.02279 to 0.02207, saving model to models/nvidia.48-0.0221
Epoch 49/100
58/58 [==============================] - 20s 339ms/step - loss: 0.0019 - val_loss: 0.0237

Epoch 00049: val_loss did not improve from 0.02207
Epoch 50/100
57/58 [============================>.] - ETA: 0s - loss: 0.0017
Epoch 00049: val_loss did not improve from 0.02207
Epoch 50/100
58/58 [==============================] - 20s 341ms/step - loss: 0.0017 - val_loss: 0.0246

Epoch 00050: val_loss did not improve from 0.02207
Epoch 51/100
57/58 [============================>.] - ETA: 0s - loss: 0.0016
Epoch 00050: val_loss did not improve from 0.02207
Epoch 51/100
58/58 [==============================] - 20s 340ms/step - loss: 0.0016 - val_loss: 0.0236

Epoch 00051: val_loss did not improve from 0.02207
Epoch 52/100
```

```
57/58 [============================>.] - ETA: 0s - loss: 0.0017
Epoch 00051: val_loss did not improve from 0.02207
Epoch 52/100
58/58 [==============================] - 20s 351ms/step - loss: 0.0017 - val_loss: 0.0242

Epoch 00052: val_loss did not improve from 0.02207
Epoch 53/100
58/58 [==============================] - 20s 349ms/step - loss: 0.0019 - val_loss: 0.0243

Epoch 00053: val_loss did not improve from 0.02207
Epoch 54/100
58/58 [==============================] - 19s 336ms/step - loss: 0.0022 - val_loss: 0.0235

Epoch 00054: val_loss did not improve from 0.02207
Epoch 55/100
57/58 [============================>.] - ETA: 0s - loss: 0.0018
Epoch 00054: val_loss did not improve from 0.02207
Epoch 55/100
58/58 [==============================] - 20s 350ms/step - loss: 0.0018 - val_loss: 0.0233

Epoch 00055: val_loss did not improve from 0.02207
Epoch 56/100
58/58 [==============================] - 20s 340ms/step - loss: 0.0016 - val_loss: 0.0250

Epoch 00056: val_loss did not improve from 0.02207
Epoch 57/100
58/58 [==============================] - 20s 350ms/step - loss: 0.0019 - val_loss: 0.0252

Epoch 00057: val_loss did not improve from 0.02207
Epoch 58/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0018 - val_loss: 0.0255

Epoch 00058: val_loss did not improve from 0.02207
Epoch 59/100
57/58 [============================>.] - ETA: 0s - loss: 0.0015
Epoch 00058: val_loss did not improve from 0.02207
Epoch 59/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0015 - val_loss: 0.0247

Epoch 00059: val_loss did not improve from 0.02207
Epoch 60/100
58/58 [==============================] - 20s 343ms/step - loss: 0.0016 - val_loss: 0.0241

Epoch 00060: val_loss did not improve from 0.02207
Epoch 61/100
58/58 [==============================] - 20s 352ms/step - loss: 0.0017 - val_loss: 0.0238

Epoch 00061: val_loss did not improve from 0.02207
```

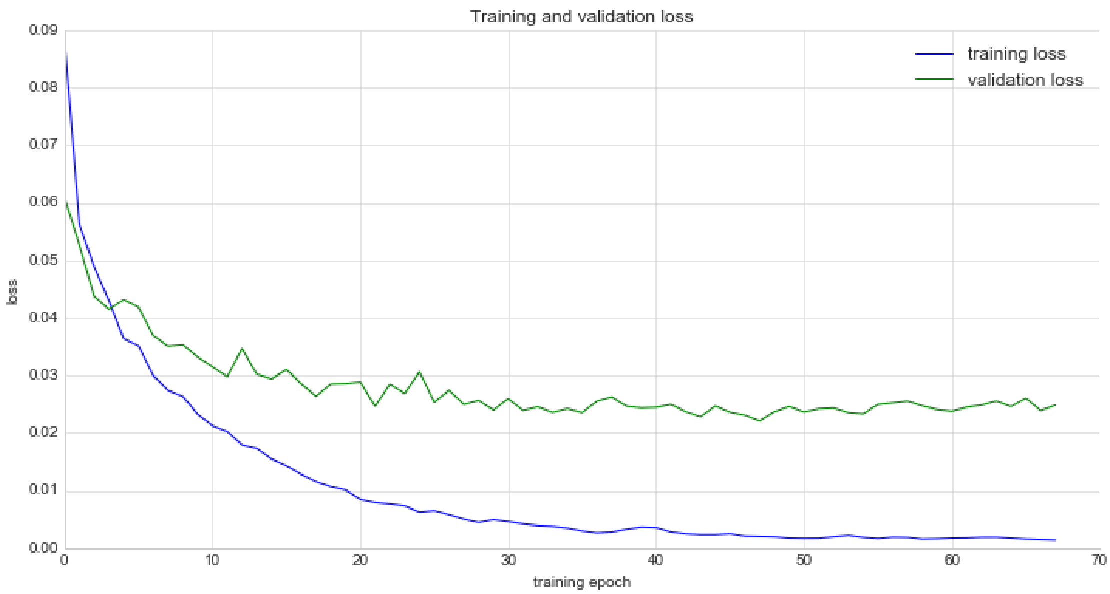
```
Epoch 62/100
58/58 [==============================] - 20s 349ms/step - loss: 0.0017 - val_loss: 0.0245

Epoch 00062: val_loss did not improve from 0.02207
Epoch 63/100
58/58 [==============================] - 21s 363ms/step - loss: 0.0018 - val_loss: 0.0248

Epoch 00063: val_loss did not improve from 0.02207
Epoch 64/100
58/58 [==============================] - 20s 350ms/step - loss: 0.0018 - val_loss: 0.0255

Epoch 00064: val_loss did not improve from 0.02207
Epoch 65/100
57/58 [============================>.] - ETA: 0s - loss: 0.0017
Epoch 00064: val_loss did not improve from 0.02207
Epoch 65/100
58/58 [==============================] - 20s 351ms/step - loss: 0.0017 - val_loss: 0.0246

Epoch 00065: val_loss did not improve from 0.02207
Epoch 66/100
58/58 [==============================] - 20s 342ms/step - loss: 0.0015 - val_loss: 0.0260

Epoch 00066: val_loss did not improve from 0.02207
Epoch 67/100
57/58 [============================>.] - ETA: 0s - loss: 0.0014
Epoch 00066: val_loss did not improve from 0.02207
Epoch 67/100
58/58 [==============================] - 20s 351ms/step - loss: 0.0014 - val_loss: 0.0239

Epoch 00067: val_loss did not improve from 0.02207
Epoch 68/100
58/58 [==============================] - 20s 343ms/step - loss: 0.0013 - val_loss: 0.0248

Epoch 00068: val_loss did not improve from 0.02207
Epoch 00068: early stopping
```

When looking at the training and validation losses, we find that both appeared to still decrease at the time training was early stopped. Training loss is still a fair amount from zero and validation loss didn't go up, so overfitting doesn't seem to be an issue yet. We might want to train for more epochs (e.g. using more patience for early stopping) and possibly use a lower learning rate.

```
In [179]: def plot_training_history(hist):
              f, ax = plt.subplots(nrows=1, ncols=1, figsize=(12, 6))
              ax.plot(hist.history['loss'])
              ax.plot(hist.history['val_loss'])
              ax.set_title('Training and validation loss')
              ax.set_ylabel('loss')
```

```
        ax.set_xlabel('training epoch')
        ax.legend(['training loss', 'validation loss'], loc='upper right')
        sns.despine()
```

```
In [50]: plot_training_history(hist)
```



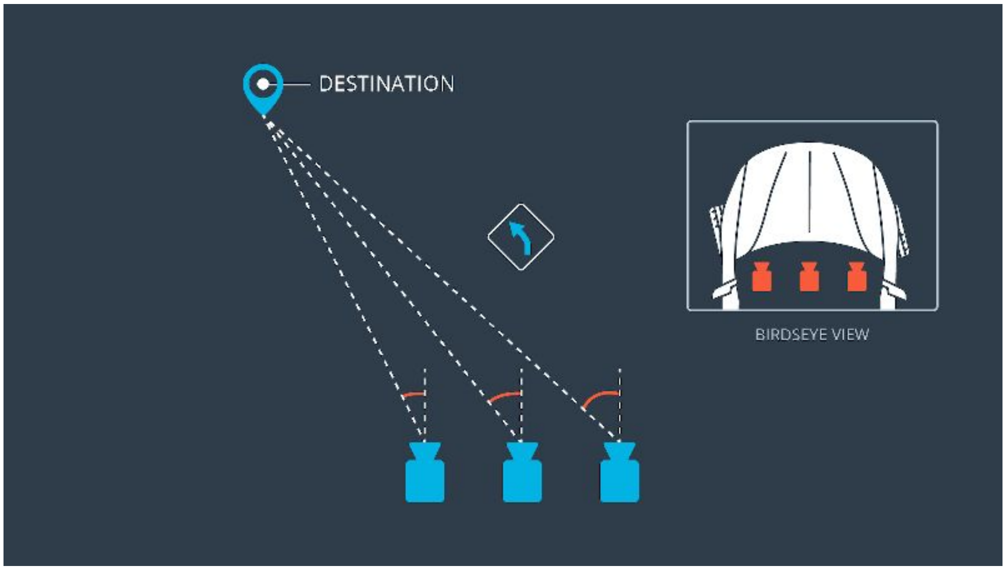

```
In [51]: def free_gpu_memory():
             K.clear_session()
             cuda.select_device(0)
             cuda.close()

         # free_gpu_memory()
```

See video/nvidia.48-0.0221.mp4 for a video of the learned autonomous steering.

0.3 Training data augmentation

Some of the inferred steering angles in the video above are a bit problematic; ignoring the fact that my driving instructions on the simulator weren't too smooth either, we can try adding more data and a somewhat noisy steering angle to force the network to generalize more.

A simple augmentation we can do is randomly flipping the images and inverting the steering angle. Another trick proposed by the NVIDIA paper is to make use of two additional cameras installed to the left and right of the car's center. This basically simulates slightly offset car positions, introducing more training data. If we assume a destination angle as determined by the center camera, due to trigonometry, we need to add correction factor to the angle whenever we're using either the left or right image, positive and negative respectively. Since no reference values for distances are given, we'll be simply using an arbitrarily selected value of e.g. `0.2`.



To prevent overfitting, a zero-mean gaussian noise is added to the recorded steering angles during training additionally.

```
In [45]: class AugmentingDataGenerator(DataGenerator):
             'Generates data for Keras'
             def __init__(self, df: pd.DataFrame, batch_size: int=32, dim: Tuple[int, int]=(320
                          dataset_path: str=DATASET_PATH, steering_std: float=0, shift_correctio
                 super().__init__(df, batch_size, dim, shuffle, dataset_path)
                 self.steering_std = steering_std
                 self.shift_correction = shift_correction

             def _get_example(self, row) -> Tuple[np.ndarray, np.float32]:
                 i = random.randint(0, 2)
                 r = random.randint(0, 1)
                 if i == 0:
                     path = os.path.join(self.dataset_path, row['Center Image'])
                     correction = 0
                 elif i == 1:
                     path = os.path.join(self.dataset_path, row['Left Image'])
                     correction = +self.shift_correction
                 else:
                     path = os.path.join(self.dataset_path, row['Right Image'])
                     correction = -self.shift_correction

                 angle = row['Steering Angle'] + correction
                 angle += np.random.normal(scale=self.steering_std)

                 img   = Image.open(path).convert('RGB')
                 if r == 1:
                     img = img.transpose(Image.FLIP_LEFT_RIGHT)
                     angle = -angle
                 X = np.array(img).astype(np.uint8)
```

```
            y = angle
            return X, y
```

With this, we restart the training.

```
In [48]: K.clear_session()
         model = nvidia_model(LEARNING_RATE)

         training_generator   = AugmentingDataGenerator(df_train, batch_size=BATCH_SIZE,
                                                         steering_std=steering_std,
                                                         shift_correction=0.2)
         validation_generator = DataGenerator(df_valid, batch_size=BATCH_SIZE)

         CHECKPOINT_PATH = os.path.join(MODEL_PATH, 'nvidia-aug.{epoch:02d}-{val_loss:.4f}.h5')
         checkpoint = ModelCheckpoint(CHECKPOINT_PATH, monitor = 'val_loss', verbose=1,
                                      save_best_only=True, mode='min')

         early_stopping = EarlyStopping(monitor='val_loss', patience=20, verbose=1,
                                        mode='min', restore_best_weights=False)

         hist = model.fit_generator(generator=training_generator,
                                    validation_data=validation_generator,
                                    use_multiprocessing=True, workers=6,
                                    callbacks=[checkpoint, early_stopping],
                                    epochs=EPOCHS, verbose=1)
```

```
Epoch 1/100
58/58 [==============================] - 21s 370ms/step - loss: 0.2108 - val_loss: 0.0917

Epoch 00001: val_loss improved from inf to 0.09166, saving model to models/nvidia-aug.01-0.091
Epoch 2/100
58/58 [==============================] - 19s 336ms/step - loss: 0.1837 - val_loss: 0.0905

Epoch 00002: val_loss improved from 0.09166 to 0.09051, saving model to models/nvidia-aug.02-0
Epoch 3/100
58/58 [==============================] - 19s 333ms/step - loss: 0.1810 - val_loss: 0.0746

Epoch 00003: val_loss improved from 0.09051 to 0.07458, saving model to models/nvidia-aug.03-0
Epoch 4/100
58/58 [==============================] - 19s 329ms/step - loss: 0.1677 - val_loss: 0.0753

Epoch 00004: val_loss did not improve from 0.07458
Epoch 5/100
58/58 [==============================] - 20s 353ms/step - loss: 0.1739 - val_loss: 0.0655

Epoch 00005: val_loss improved from 0.07458 to 0.06546, saving model to models/nvidia-aug.05-0
Epoch 6/100
57/58 [============================>.] - ETA: 0s - loss: 0.1674
```

```
Epoch 00005: val_loss improved from 0.07458 to 0.06546, saving model to models/nvidia-aug.05-0
58/58 [==============================] - 21s 357ms/step - loss: 0.1668 - val_loss: 0.0654

Epoch 00006: val_loss improved from 0.06546 to 0.06540, saving model to models/nvidia-aug.06-0
Epoch 7/100
58/58 [==============================] - 20s 347ms/step - loss: 0.1587 - val_loss: 0.0603

Epoch 00007: val_loss improved from 0.06540 to 0.06026, saving model to models/nvidia-aug.07-0
Epoch 8/100
58/58 [==============================] - 21s 361ms/step - loss: 0.1522 - val_loss: 0.0508

Epoch 00008: val_loss improved from 0.06026 to 0.05079, saving model to models/nvidia-aug.08-0
Epoch 9/100
58/58 [==============================] - 20s 338ms/step - loss: 0.1552 - val_loss: 0.0512

Epoch 00009: val_loss did not improve from 0.05079
Epoch 10/100
57/58 [============================>.] - ETA: 0s - loss: 0.1581
Epoch 00009: val_loss did not improve from 0.05079
Epoch 10/100
58/58 [==============================] - 20s 345ms/step - loss: 0.1589 - val_loss: 0.0597

Epoch 00010: val_loss did not improve from 0.05079
Epoch 11/100
58/58 [==============================] - 21s 362ms/step - loss: 0.1539 - val_loss: 0.0494

Epoch 00011: val_loss improved from 0.05079 to 0.04937, saving model to models/nvidia-aug.11-0
Epoch 12/100
58/58 [==============================] - 20s 345ms/step - loss: 0.1597 - val_loss: 0.0511

Epoch 00012: val_loss did not improve from 0.04937
Epoch 13/100
58/58 [==============================] - 20s 350ms/step - loss: 0.1489 - val_loss: 0.0493

Epoch 00013: val_loss improved from 0.04937 to 0.04932, saving model to models/nvidia-aug.13-0
Epoch 14/100
58/58 [==============================] - 20s 350ms/step - loss: 0.1487 - val_loss: 0.0487

Epoch 00014: val_loss improved from 0.04932 to 0.04869, saving model to models/nvidia-aug.14-0
Epoch 15/100
58/58 [==============================] - 20s 350ms/step - loss: 0.1540 - val_loss: 0.0473

Epoch 00015: val_loss improved from 0.04869 to 0.04735, saving model to models/nvidia-aug.15-0
Epoch 16/100
58/58 [==============================] - 20s 349ms/step - loss: 0.1485 - val_loss: 0.0518

Epoch 00016: val_loss did not improve from 0.04735
Epoch 17/100
```

```
58/58 [==============================] - 20s 349ms/step - loss: 0.1434 - val_loss: 0.0483

Epoch 00017: val_loss did not improve from 0.04735
Epoch 18/100
57/58 [============================>.] - ETA: 0s - loss: 0.1403
Epoch 00017: val_loss did not improve from 0.04735
Epoch 18/100
58/58 [==============================] - 20s 344ms/step - loss: 0.1405 - val_loss: 0.0456

Epoch 00018: val_loss improved from 0.04735 to 0.04563, saving model to models/nvidia-aug.18-0
Epoch 19/100
57/58 [============================>.] - ETA: 0s - loss: 0.1384
Epoch 00018: val_loss improved from 0.04735 to 0.04563, saving model to models/nvidia-aug.18-0
58/58 [==============================] - 20s 337ms/step - loss: 0.1387 - val_loss: 0.0457

Epoch 00019: val_loss did not improve from 0.04563
Epoch 20/100
58/58 [==============================] - 20s 345ms/step - loss: 0.1510 - val_loss: 0.0472

Epoch 00020: val_loss did not improve from 0.04563
Epoch 21/100
57/58 [============================>.] - ETA: 0s - loss: 0.1386
Epoch 00020: val_loss did not improve from 0.04563
Epoch 21/100
58/58 [==============================] - 21s 362ms/step - loss: 0.1386 - val_loss: 0.0459

Epoch 00021: val_loss did not improve from 0.04563
Epoch 22/100
58/58 [==============================] - 21s 354ms/step - loss: 0.1461 - val_loss: 0.0486

Epoch 00022: val_loss did not improve from 0.04563
Epoch 23/100
58/58 [==============================] - 20s 348ms/step - loss: 0.1390 - val_loss: 0.0480

Epoch 00023: val_loss did not improve from 0.04563
Epoch 24/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1432 - val_loss: 0.0417

Epoch 00024: val_loss improved from 0.04563 to 0.04167, saving model to models/nvidia-aug.24-0
Epoch 25/100
58/58 [==============================] - 19s 335ms/step - loss: 0.1399 - val_loss: 0.0436

Epoch 00025: val_loss did not improve from 0.04167
Epoch 26/100
57/58 [============================>.] - ETA: 0s - loss: 0.1407
Epoch 00025: val_loss did not improve from 0.04167
Epoch 26/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1408 - val_loss: 0.0421
```

```
Epoch 00026: val_loss did not improve from 0.04167
Epoch 27/100
58/58 [==============================] - 20s 343ms/step - loss: 0.1372 - val_loss: 0.0395

Epoch 00027: val_loss improved from 0.04167 to 0.03951, saving model to models/nvidia-aug.27-0
Epoch 28/100
58/58 [==============================] - 20s 351ms/step - loss: 0.1415 - val_loss: 0.0420

Epoch 00028: val_loss did not improve from 0.03951
Epoch 29/100
58/58 [==============================] - 20s 353ms/step - loss: 0.1406 - val_loss: 0.0377

Epoch 00029: val_loss improved from 0.03951 to 0.03773, saving model to models/nvidia-aug.29-0
Epoch 30/100
57/58 [============================>.] - ETA: 0s - loss: 0.1403
Epoch 00029: val_loss improved from 0.03951 to 0.03773, saving model to models/nvidia-aug.29-0
58/58 [==============================] - 22s 373ms/step - loss: 0.1401 - val_loss: 0.0381

Epoch 00030: val_loss did not improve from 0.03773
Epoch 31/100
58/58 [==============================] - 22s 372ms/step - loss: 0.1377 - val_loss: 0.0350

Epoch 00031: val_loss improved from 0.03773 to 0.03500, saving model to models/nvidia-aug.31-0
Epoch 32/100
58/58 [==============================] - 20s 353ms/step - loss: 0.1302 - val_loss: 0.0377

Epoch 00032: val_loss did not improve from 0.03500
Epoch 33/100
58/58 [==============================] - 21s 357ms/step - loss: 0.1401 - val_loss: 0.0435

Epoch 00033: val_loss did not improve from 0.03500
Epoch 34/100
58/58 [==============================] - 21s 366ms/step - loss: 0.1345 - val_loss: 0.0375

Epoch 00034: val_loss did not improve from 0.03500
Epoch 35/100
58/58 [==============================] - 20s 343ms/step - loss: 0.1328 - val_loss: 0.0344

Epoch 00035: val_loss improved from 0.03500 to 0.03443, saving model to models/nvidia-aug.35-0
Epoch 36/100
58/58 [==============================] - 21s 363ms/step - loss: 0.1283 - val_loss: 0.0375

Epoch 00036: val_loss did not improve from 0.03443
Epoch 37/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1285 - val_loss: 0.0375

Epoch 00037: val_loss did not improve from 0.03443
```

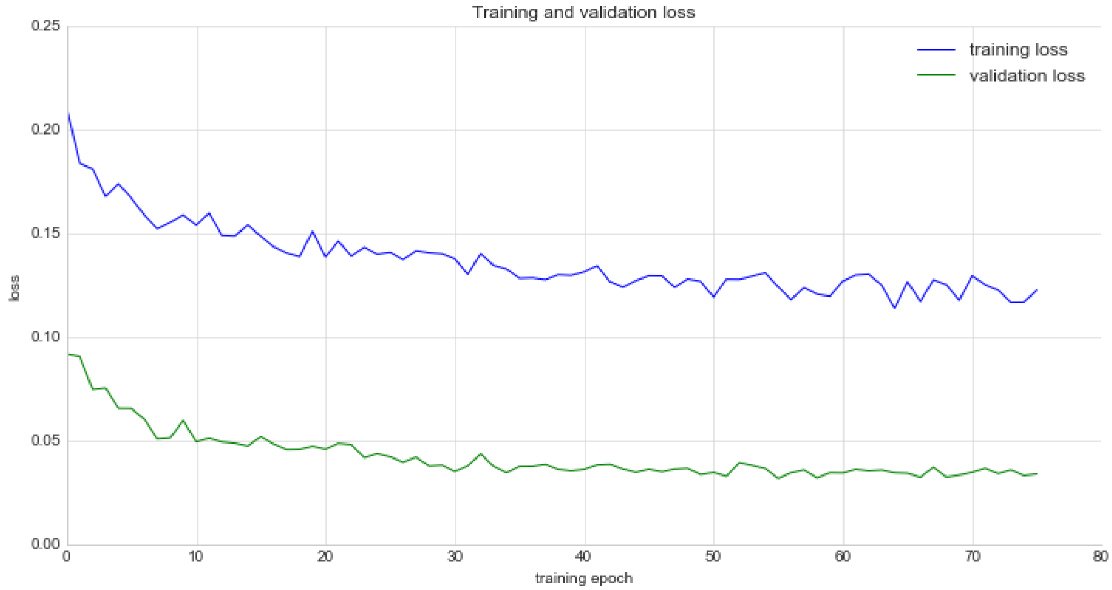
```
Epoch 38/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1276 - val_loss: 0.0385

Epoch 00038: val_loss did not improve from 0.03443
Epoch 39/100
58/58 [==============================] - 20s 348ms/step - loss: 0.1300 - val_loss: 0.0360

Epoch 00039: val_loss did not improve from 0.03443
Epoch 40/100
58/58 [==============================] - 20s 348ms/step - loss: 0.1297 - val_loss: 0.0353

Epoch 00040: val_loss did not improve from 0.03443
Epoch 41/100
57/58 [============================>.] - ETA: 0s - loss: 0.1313
Epoch 00040: val_loss did not improve from 0.03443
Epoch 41/100
58/58 [==============================] - 20s 349ms/step - loss: 0.1314 - val_loss: 0.0360

Epoch 00041: val_loss did not improve from 0.03443
Epoch 42/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1343 - val_loss: 0.0382

Epoch 00042: val_loss did not improve from 0.03443
Epoch 43/100
58/58 [==============================] - 21s 361ms/step - loss: 0.1266 - val_loss: 0.0385

Epoch 00043: val_loss did not improve from 0.03443
Epoch 44/100
57/58 [============================>.] - ETA: 0s - loss: 0.1239
58/58 [==============================] - 21s 359ms/step - loss: 0.1240 - val_loss: 0.0362

Epoch 00044: val_loss did not improve from 0.03443
Epoch 45/100
58/58 [==============================] - 20s 340ms/step - loss: 0.1271 - val_loss: 0.0346

Epoch 00045: val_loss did not improve from 0.03443
Epoch 46/100
58/58 [==============================] - 20s 347ms/step - loss: 0.1295 - val_loss: 0.0361

Epoch 00046: val_loss did not improve from 0.03443
Epoch 47/100
58/58 [==============================] - 20s 339ms/step - loss: 0.1294 - val_loss: 0.0349

Epoch 00047: val_loss did not improve from 0.03443
Epoch 48/100
58/58 [==============================] - 20s 349ms/step - loss: 0.1240 - val_loss: 0.0362

Epoch 00048: val_loss did not improve from 0.03443
```

```
Epoch 49/100
58/58 [==============================] - 22s 371ms/step - loss: 0.1279 - val_loss: 0.0365

Epoch 00049: val_loss did not improve from 0.03443
Epoch 50/100
58/58 [==============================] - 22s 377ms/step - loss: 0.1268 - val_loss: 0.0335

Epoch 00050: val_loss improved from 0.03443 to 0.03355, saving model to models/nvidia-aug.50-0
Epoch 51/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1192 - val_loss: 0.0345

Epoch 00051: val_loss did not improve from 0.03355
Epoch 52/100
58/58 [==============================] - 20s 341ms/step - loss: 0.1278 - val_loss: 0.0328

Epoch 00052: val_loss improved from 0.03355 to 0.03279, saving model to models/nvidia-aug.52-0
Epoch 53/100
58/58 [==============================] - 20s 340ms/step - loss: 0.1277 - val_loss: 0.0393

Epoch 00053: val_loss did not improve from 0.03279
Epoch 54/100
58/58 [==============================] - 20s 348ms/step - loss: 0.1293 - val_loss: 0.0380

Epoch 00054: val_loss did not improve from 0.03279
Epoch 55/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1309 - val_loss: 0.0365

Epoch 00055: val_loss did not improve from 0.03279
Epoch 56/100
57/58 [============================>.] - ETA: 0s - loss: 0.1245
Epoch 00055: val_loss did not improve from 0.03279
Epoch 56/100
58/58 [==============================] - 20s 344ms/step - loss: 0.1241 - val_loss: 0.0317

Epoch 00056: val_loss improved from 0.03279 to 0.03175, saving model to models/nvidia-aug.56-0
Epoch 57/100
57/58 [============================>.] - ETA: 0s - loss: 0.1179
Epoch 00056: val_loss improved from 0.03279 to 0.03175, saving model to models/nvidia-aug.56-0
58/58 [==============================] - 20s 349ms/step - loss: 0.1180 - val_loss: 0.0344

Epoch 00057: val_loss did not improve from 0.03175
Epoch 58/100
58/58 [==============================] - 20s 344ms/step - loss: 0.1238 - val_loss: 0.0358

Epoch 00058: val_loss did not improve from 0.03175
Epoch 59/100
58/58 [==============================] - 20s 339ms/step - loss: 0.1206 - val_loss: 0.0320
```

```
Epoch 00059: val_loss did not improve from 0.03175
Epoch 60/100
57/58 [============================>.] - ETA: 0s - loss: 0.1198
58/58 [==============================] - 20s 351ms/step - loss: 0.1196 - val_loss: 0.0344

Epoch 00060: val_loss did not improve from 0.03175
Epoch 61/100
58/58 [==============================] - 20s 345ms/step - loss: 0.1269 - val_loss: 0.0343

Epoch 00061: val_loss did not improve from 0.03175
Epoch 62/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1298 - val_loss: 0.0360

Epoch 00062: val_loss did not improve from 0.03175
Epoch 63/100
58/58 [==============================] - 20s 338ms/step - loss: 0.1302 - val_loss: 0.0353

Epoch 00063: val_loss did not improve from 0.03175
Epoch 64/100
58/58 [==============================] - 20s 338ms/step - loss: 0.1302 - val_loss: 0.0353
58/58 [==============================] - 20s 345ms/step - loss: 0.1249 - val_loss: 0.0357

Epoch 00064: val_loss did not improve from 0.03175
Epoch 65/100
57/58 [============================>.] - ETA: 0s - loss: 0.1142
Epoch 00064: val_loss did not improve from 0.03175
Epoch 65/100
58/58 [==============================] - 20s 350ms/step - loss: 0.1139 - val_loss: 0.0344

Epoch 00065: val_loss did not improve from 0.03175
Epoch 66/100
58/58 [==============================] - 20s 338ms/step - loss: 0.1266 - val_loss: 0.0342

Epoch 00066: val_loss did not improve from 0.03175
Epoch 67/100
58/58 [==============================] - 20s 344ms/step - loss: 0.1170 - val_loss: 0.0322

Epoch 00067: val_loss did not improve from 0.03175
Epoch 68/100
58/58 [==============================] - 20s 352ms/step - loss: 0.1275 - val_loss: 0.0371

Epoch 00068: val_loss did not improve from 0.03175
Epoch 69/100
58/58 [==============================] - 20s 342ms/step - loss: 0.1251 - val_loss: 0.0324

Epoch 00069: val_loss did not improve from 0.03175
Epoch 70/100
57/58 [============================>.] - ETA: 0s - loss: 0.1175
```

```
Epoch 00069: val_loss did not improve from 0.03175
Epoch 70/100
58/58 [==============================] - 20s 348ms/step - loss: 0.1177 - val_loss: 0.0333

Epoch 00070: val_loss did not improve from 0.03175
Epoch 71/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1294 - val_loss: 0.0346

Epoch 00071: val_loss did not improve from 0.03175
Epoch 72/100
57/58 [============================>.] - ETA: 0s - loss: 0.1250
Epoch 00071: val_loss did not improve from 0.03175
Epoch 72/100
58/58 [==============================] - 20s 352ms/step - loss: 0.1251 - val_loss: 0.0366

Epoch 00072: val_loss did not improve from 0.03175
Epoch 73/100
58/58 [==============================] - 20s 346ms/step - loss: 0.1226 - val_loss: 0.0340

Epoch 00073: val_loss did not improve from 0.03175
Epoch 74/100
58/58 [==============================] - 20s 347ms/step - loss: 0.1167 - val_loss: 0.0357

Epoch 00074: val_loss did not improve from 0.03175
Epoch 75/100
57/58 [============================>.] - ETA: 0s - loss: 0.1169
Epoch 00074: val_loss did not improve from 0.03175
Epoch 75/100
58/58 [==============================] - 21s 354ms/step - loss: 0.1167 - val_loss: 0.0331

Epoch 00075: val_loss did not improve from 0.03175
Epoch 76/100
58/58 [==============================] - 20s 342ms/step - loss: 0.1226 - val_loss: 0.0338

Epoch 00076: val_loss did not improve from 0.03175
Epoch 00076: early stopping

Epoch 00076: val_loss did not improve from 0.03175
```

```
In [49]: plot_training_history(hist)
```



Interestingly - due to the noise augmentation - the validation loss is now much lower than the training loss. Even in comparison to the previous experiment the validation loss is now higher; however, a visual inspection of the driving results show that the car is now driving much more in the center of the lane, rather than sticking to the sides.

Here are some videos from autonomous driving mode with network controls:

See `video/nvidia-aug.56-0.0317.mp4` for one lap on the training track (bigger version on YouTube).

Here's `video/nvidia-aug-2.56-0.0317.mp4` for some turns on the harder track with an unintended stop (bigger version on YouTube).

And another slightly longer run on the same track in `video/nvidia-aug-2_longer.56-0.0317.mp4`.

The accuracy on the harder track is suprisingly high, however the network fails to safely maneuver the car all the way. Sampling more data from there would certainly help.

0.4 Alternative network architecture

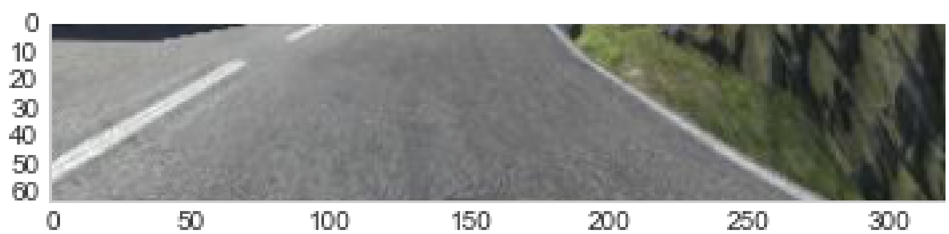
From the training and validation loss plots, we don't have reason to believe the network has overfit the training data at all. However, the project rubrik requires using additional countermeasures such as dropout or regularization. Since this has to be added anyway, we can directly experiment with different network architectures as well.

We'll also be feeding in the original image size and crop the entire top part.

```
In [150]: test_img_path = os.path.join(DATASET_PATH, df_train.iloc[1]['Center Image'])
          test_img = Image.open(test_img_path).convert('RGB')
          test_img = np.array(test_img).astype(np.float32) / 255.
          plt.imshow(test_img)
          plt.grid(None)
          sns.despine()
```


```
In [151]: CROP_TOP_BIG = 75
          CROP_BOTTOM_BIG = 22

          cropped_test_img = test_img[CROP_TOP_BIG:-CROP_BOTTOM_BIG, ...]
          print('Cropped image size: {}'.format(cropped_test_img.shape))
          plt.imshow(cropped_test_img)
          plt.grid(None)
          sns.despine()
```

```
Cropped image size: (63, 320, 3)
```



The core changes we'll be doing to the NVIDIA network are the following:

- The original image size is used, i.e. resizing is removed,
- 2D convolutions are replaced with depthwise separable 2D convolutions,
- ReLU activations are replaced with Parametric ReLU, and
- Dropout is added before the first fully connected layer.

The removal of the resizing step is done purely to improve processing speed; at the same time, the introduction of depthwise separable convolutions should allow for some reduction of parameters as well. Much more importantly, using a depthwise separable convolution in the input layer follows the intuition, that grayscale intensity matters much more to lane detection that color. By learning filters individually for the luminance (Y) and chrominance (U and V) channels, it should be possible to focus the training on the more important channels rather than trying to use them simultaneously. Separable convolutions are kept for later layers because of the same reason; a rather efficient class of neural networks designed for mobile use (such as embedded systems) using separable convolutions are MobileNets.

ReLU activations drop all negative activations, effectively cancelling out their gradient, possibly leading to “dead” neurons during training (when a neuron’s gradient is zero, no learning can happen ever again). While this allows for pruning of the network, it possibly throws aways computational capabilities of the architecture. Hence, Parametric ReLU units are used instead: Functioning like Leaky ReLU, a class of activation functions that “leaks” negative activations by using a small coefficient on them, Parametric ReLUs allow for learning the influence of negative activations as well at the cost of one extra trainable parameter per activation function. Nowadays, activation functions such as ELU and SELU have proven to be more efficient in terms of training and inference quality; they do rely on exponential functions though, and since the goal is to have a network that should run as fast as possible on an embedded device, I decided not to use them.

Lastly, dropout is applied exactly once. Since dropout randomly “drops” connections during training, the network has to learn redundancy. When repeatedly using dropouts in subsequent layers, this only becomes worse. Again, pruning may mitigate this effect by simply removing “identical” neurons, but this is beyond the scope of this project, more than one instance of dropout in the network was deemed enough.

```
In [95]: from keras.layers import SeparableConv2D, Dropout, PReLU
```

```
In [175]: def custom_model(learning_rate: float, decay: float=0, dropout_rate: float=0.3):
              model = Sequential()

              # YUV conversion and normalization
              model.add(Lambda(yuv_conversion,  input_shape = (160, 320, 3), name='rgb_to_yuv'
              model.add(Lambda(lambda x: x * 2 - 1, name='normalize'))

              # Crop the image to remove the sky and car hood
              model.add(Cropping2D(cropping=((CROP_TOP_BIG, CROP_BOTTOM_BIG), (0, 0)),
                                   name='crop'))

              model.add(SeparableConv2D(24, (5, 5), strides=(3, 3), padding='valid',
                                        depth_multiplier=16, name='conv_1'))
              model.add(PReLU(name='conv_1_prelu'))

              model.add(SeparableConv2D(36, (5, 5), strides=(3, 3), padding='valid',
                                        depth_multiplier=9, name='conv_2'))
              model.add(PReLU(name='conv_2_prelu'))

              model.add(SeparableConv2D(48, (3, 3), strides=(2, 2), padding='valid',
                                        depth_multiplier=6, name='conv_3'))
```

```
        model.add(PReLU(name='conv_3_prelu'))

        model.add(SeparableConv2D(64, (2, 2), strides=(2, 2), padding='valid',
                                  depth_multiplier=9, name='conv_4'))
        model.add(PReLU(name='conv_4_prelu'))

        model.add(Flatten())
        model.add(Dropout(rate=dropout_rate))

        model.add(Dense(100, name='fc_1'))
        model.add(PReLU(name='fc_1_prelu'))

        model.add(Dense(50, name='fc_2'))
        model.add(PReLU(name='fc_2_prelu'))

        model.add(Dense(10, name='fc_3'))
        model.add(PReLU(name='fc_3_prelu'))

        model.add(Dense(1, name='angle'))

        adam = Adam(lr=learning_rate, decay=decay)
        model.compile(optimizer=adam, loss='mse')

        return model

    K.clear_session()
    model = custom_model(LEARNING_RATE)
    model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rgb_to_yuv (Lambda)          (None, 160, 320, 3)       0
_________________________________________________________________
normalize (Lambda)           (None, 160, 320, 3)       0
_________________________________________________________________
crop (Cropping2D)            (None, 63, 320, 3)        0
_________________________________________________________________
conv_1 (SeparableConv2D)     (None, 20, 106, 24)       2376
_________________________________________________________________
conv_1_prelu (PReLU)         (None, 20, 106, 24)       50880
_________________________________________________________________
conv_2 (SeparableConv2D)     (None, 6, 34, 36)         13212
_________________________________________________________________
conv_2_prelu (PReLU)         (None, 6, 34, 36)         7344
_________________________________________________________________
conv_3 (SeparableConv2D)     (None, 2, 16, 48)         12360
_________________________________________________________________
```

```
conv_3_prelu (PReLU)         (None, 2, 16, 48)         1536
_________________________________________________________________
conv_4 (SeparableConv2D)     (None, 1, 8, 64)          29440
_________________________________________________________________
conv_4_prelu (PReLU)         (None, 1, 8, 64)          512
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
fc_1 (Dense)                 (None, 100)               51300
_________________________________________________________________
fc_1_prelu (PReLU)           (None, 100)               100
_________________________________________________________________
fc_2 (Dense)                 (None, 50)                5050
_________________________________________________________________
fc_2_prelu (PReLU)           (None, 50)                50
_________________________________________________________________
fc_3 (Dense)                 (None, 10)                510
_________________________________________________________________
fc_3_prelu (PReLU)           (None, 10)                10
_________________________________________________________________
angle (Dense)                (None, 1)                 11
=================================================================
Total params: 174,691
Trainable params: 174,691
Non-trainable params: 0
_________________________________________________________________
```

So far, we use less parameters than the original network, allowing for faster inference.

In addition, we'll be explicitly instantiating the images from the left and right cameras as first-class training examples (as opposed to be before, where they were treated as on-line augmentations).

```
In [176]: class AugmentingDataGenerator2(DataGenerator):
              'Generates data for Keras'
              def __init__(self, df: pd.DataFrame, batch_size: int=32, dim: Tuple[int, int]=(3
                           dataset_path: str=DATASET_PATH, steering_std: float=0, shift_correct

                  # Combine the training examples
                  df_c = df[['Center Image', 'Steering Angle']].rename(columns={'Center Image'
                  df_l = df[['Left Image', 'Steering Angle']].rename(columns={'Left Image': 'Im
                  df_r = df[['Right Image', 'Steering Angle']].rename(columns={'Right Image':
                  df_l['Steering Angle'] += shift_correction
                  df_r['Steering Angle'] -= shift_correction
                  df = pd.concat([df_c, df_l, df_r], axis=0, sort=False)
```

```
        super().__init__(df, batch_size, dim, shuffle, dataset_path)
        self.steering_std = steering_std
        self.shift_correction = shift_correction

    def _get_example(self, row) -> Tuple[np.ndarray, np.float32]:
        path = os.path.join(self.dataset_path, row['Image'])
        angle = row['Steering Angle']
        angle += np.random.normal(scale=self.steering_std)

        img = Image.open(path).convert('RGB')
        if random.choice([True, False]):
            img = img.transpose(Image.FLIP_LEFT_RIGHT)
            angle = -angle
        X = np.array(img).astype(np.uint8)
        y = angle
        return X, y
```

We now run the training again.

```
In [177]: K.clear_session()
          model = custom_model(LEARNING_RATE)

          training_generator   = AugmentingDataGenerator2(df_train, batch_size=BATCH_SIZE,
                                                          steering_std=steering_std / 2,
                                                          shift_correction=0.2)
          validation_generator = DataGenerator(df_valid, batch_size=BATCH_SIZE)

          CHECKPOINT_PATH = os.path.join(MODEL_PATH, 'custom-1.{epoch:02d}-{val_loss:.4f}.h5')
          checkpoint = ModelCheckpoint(CHECKPOINT_PATH, monitor = 'val_loss', verbose=1,
                                       save_best_only=True, mode='min')

          early_stopping = EarlyStopping(monitor='val_loss', patience=20, verbose=1,
                                         mode='min', restore_best_weights=False)

          hist = model.fit_generator(generator=training_generator,
                                     validation_data=validation_generator,
                                     use_multiprocessing=True, workers=6,
                                     callbacks=[checkpoint, early_stopping],
                                     epochs=EPOCHS, verbose=1)
```

```
Epoch 1/100
175/175 [==============================] - 90s 517ms/step - loss: 0.1416 - val_loss: 0.0740

Epoch 00001: val_loss improved from inf to 0.07402, saving model to models/custom-1.01-0.0740.h
Epoch 2/100
175/175 [==============================] - 94s 537ms/step - loss: 0.1066 - val_loss: 0.0601

Epoch 00002: val_loss improved from 0.07402 to 0.06011, saving model to models/custom-1.02-0.06
```

```
Epoch 3/100
175/175 [==============================] - 93s 531ms/step - loss: 0.0961 - val_loss: 0.0505

Epoch 00003: val_loss improved from 0.06011 to 0.05048, saving model to models/custom-1.03-0.05
Epoch 4/100
175/175 [==============================] - 92s 527ms/step - loss: 0.0886 - val_loss: 0.0502

Epoch 00004: val_loss improved from 0.05048 to 0.05017, saving model to models/custom-1.04-0.05
Epoch 5/100
175/175 [==============================] - 94s 539ms/step - loss: 0.0850 - val_loss: 0.0470

Epoch 00005: val_loss improved from 0.05017 to 0.04700, saving model to models/custom-1.05-0.04
Epoch 6/100
175/175 [==============================] - 94s 536ms/step - loss: 0.0813 - val_loss: 0.0454

Epoch 00006: val_loss improved from 0.04700 to 0.04540, saving model to models/custom-1.06-0.04
Epoch 7/100
175/175 [==============================] - 93s 529ms/step - loss: 0.0783 - val_loss: 0.0427

Epoch 00007: val_loss improved from 0.04540 to 0.04273, saving model to models/custom-1.07-0.04
Epoch 8/100
175/175 [==============================] - 93s 530ms/step - loss: 0.0783 - val_loss: 0.0420

Epoch 00008: val_loss improved from 0.04273 to 0.04205, saving model to models/custom-1.08-0.04
Epoch 9/100
175/175 [==============================] - 96s 549ms/step - loss: 0.0772 - val_loss: 0.0455

Epoch 00009: val_loss did not improve from 0.04205
Epoch 10/100
175/175 [==============================] - 94s 536ms/step - loss: 0.0744 - val_loss: 0.0433

Epoch 00010: val_loss did not improve from 0.04205
Epoch 11/100
175/175 [==============================] - 93s 532ms/step - loss: 0.0726 - val_loss: 0.0431

Epoch 00011: val_loss did not improve from 0.04205
Epoch 12/100
175/175 [==============================] - 95s 545ms/step - loss: 0.0742 - val_loss: 0.0397

Epoch 00012: val_loss improved from 0.04205 to 0.03966, saving model to models/custom-1.12-0.03
Epoch 13/100
175/175 [==============================] - 94s 534ms/step - loss: 0.0715 - val_loss: 0.0383

Epoch 00013: val_loss improved from 0.03966 to 0.03834, saving model to models/custom-1.13-0.03
Epoch 14/100
175/175 [==============================] - 92s 528ms/step - loss: 0.0709 - val_loss: 0.0384

Epoch 00014: val_loss did not improve from 0.03834
```

```
Epoch 15/100
175/175 [==============================] - 95s 542ms/step - loss: 0.0686 - val_loss: 0.0408

Epoch 00015: val_loss did not improve from 0.03834
Epoch 16/100
175/175 [==============================] - 95s 541ms/step - loss: 0.0710 - val_loss: 0.0399

Epoch 00016: val_loss did not improve from 0.03834
Epoch 17/100
175/175 [==============================] - 93s 531ms/step - loss: 0.0679 - val_loss: 0.0403

Epoch 00017: val_loss did not improve from 0.03834
Epoch 18/100
175/175 [==============================] - 91s 519ms/step - loss: 0.0670 - val_loss: 0.0382

Epoch 00018: val_loss improved from 0.03834 to 0.03822, saving model to models/custom-1.18-0.0
Epoch 19/100
175/175 [==============================] - 104s 595ms/step - loss: 0.0658 - val_loss: 0.0362

Epoch 00019: val_loss improved from 0.03822 to 0.03620, saving model to models/custom-1.19-0.0
Epoch 20/100
175/175 [==============================] - 99s 564ms/step - loss: 0.0659 - val_loss: 0.0365

Epoch 00020: val_loss did not improve from 0.03620
Epoch 21/100
175/175 [==============================] - 93s 531ms/step - loss: 0.0653 - val_loss: 0.0357

Epoch 00021: val_loss improved from 0.03620 to 0.03568, saving model to models/custom-1.21-0.0
Epoch 22/100
175/175 [==============================] - 91s 520ms/step - loss: 0.0639 - val_loss: 0.0351

Epoch 00022: val_loss improved from 0.03568 to 0.03510, saving model to models/custom-1.22-0.0
Epoch 23/100
175/175 [==============================] - 91s 521ms/step - loss: 0.0639 - val_loss: 0.0344

Epoch 00023: val_loss improved from 0.03510 to 0.03437, saving model to models/custom-1.23-0.0
Epoch 24/100
175/175 [==============================] - 95s 544ms/step - loss: 0.0636 - val_loss: 0.0350

Epoch 00024: val_loss did not improve from 0.03437
Epoch 25/100
175/175 [==============================] - 95s 543ms/step - loss: 0.0631 - val_loss: 0.0368

Epoch 00025: val_loss did not improve from 0.03437
Epoch 26/100
175/175 [==============================] - 97s 553ms/step - loss: 0.0629 - val_loss: 0.0352

Epoch 00026: val_loss did not improve from 0.03437
```

```
Epoch 27/100
175/175 [==============================] - 99s 568ms/step - loss: 0.0620 - val_loss: 0.0339

Epoch 00027: val_loss improved from 0.03437 to 0.03388, saving model to models/custom-1.27-0.03
Epoch 28/100
175/175 [==============================] - 94s 534ms/step - loss: 0.0616 - val_loss: 0.0377

Epoch 00028: val_loss did not improve from 0.03388
Epoch 29/100
175/175 [==============================] - 95s 545ms/step - loss: 0.0605 - val_loss: 0.0351

Epoch 00029: val_loss did not improve from 0.03388
Epoch 30/100
175/175 [==============================] - 94s 535ms/step - loss: 0.0602 - val_loss: 0.0339

Epoch 00030: val_loss did not improve from 0.03388
Epoch 31/100
175/175 [==============================] - 93s 531ms/step - loss: 0.0595 - val_loss: 0.0367

Epoch 00031: val_loss did not improve from 0.03388
Epoch 32/100
174/175 [============================>.] - ETA: 0s - loss: 0.0603
Epoch 00031: val_loss did not improve from 0.03388
Epoch 32/100
175/175 [==============================] - 94s 535ms/step - loss: 0.0602 - val_loss: 0.0332

Epoch 00032: val_loss improved from 0.03388 to 0.03323, saving model to models/custom-1.32-0.03
Epoch 33/100
175/175 [==============================] - 92s 526ms/step - loss: 0.0597 - val_loss: 0.0348

Epoch 00033: val_loss did not improve from 0.03323
Epoch 34/100
175/175 [==============================] - 94s 538ms/step - loss: 0.0594 - val_loss: 0.0333

Epoch 00034: val_loss did not improve from 0.03323
Epoch 35/100
175/175 [==============================] - 94s 537ms/step - loss: 0.0585 - val_loss: 0.0334

Epoch 00035: val_loss did not improve from 0.03323
Epoch 36/100
175/175 [==============================] - 94s 535ms/step - loss: 0.0588 - val_loss: 0.0330

Epoch 00036: val_loss improved from 0.03323 to 0.03301, saving model to models/custom-1.36-0.03
Epoch 37/100
175/175 [==============================] - 100s 572ms/step - loss: 0.0587 - val_loss: 0.0366

Epoch 00037: val_loss did not improve from 0.03301
Epoch 38/100
```

```
175/175 [==============================] - 94s 539ms/step - loss: 0.0571 - val_loss: 0.0351

Epoch 00038: val_loss did not improve from 0.03301
Epoch 39/100
175/175 [==============================] - 92s 528ms/step - loss: 0.0574 - val_loss: 0.0362

Epoch 00039: val_loss did not improve from 0.03301
Epoch 40/100
175/175 [==============================] - 102s 582ms/step - loss: 0.0575 - val_loss: 0.0314

Epoch 00040: val_loss improved from 0.03301 to 0.03144, saving model to models/custom-1.40-0.03
Epoch 41/100
175/175 [==============================] - 98s 559ms/step - loss: 0.0567 - val_loss: 0.0330

Epoch 00041: val_loss did not improve from 0.03144
Epoch 42/100
175/175 [==============================] - 91s 522ms/step - loss: 0.0573 - val_loss: 0.0326

Epoch 00042: val_loss did not improve from 0.03144
Epoch 43/100
175/175 [==============================] - 92s 524ms/step - loss: 0.0567 - val_loss: 0.0324

Epoch 00043: val_loss did not improve from 0.03144
Epoch 44/100
175/175 [==============================] - 92s 524ms/step - loss: 0.0587 - val_loss: 0.0387

Epoch 00044: val_loss did not improve from 0.03144
Epoch 45/100
175/175 [==============================] - 91s 522ms/step - loss: 0.0560 - val_loss: 0.0330

Epoch 00045: val_loss did not improve from 0.03144
Epoch 46/100
175/175 [==============================] - 92s 527ms/step - loss: 0.0563 - val_loss: 0.0324

Epoch 00046: val_loss did not improve from 0.03144
Epoch 47/100
175/175 [==============================] - 183s 1s/step - loss: 0.0550 - val_loss: 0.0320

Epoch 00047: val_loss did not improve from 0.03144
Epoch 48/100
175/175 [==============================] - 222s 1s/step - loss: 0.0548 - val_loss: 0.0345

Epoch 00048: val_loss did not improve from 0.03144
Epoch 49/100
175/175 [==============================] - 221s 1s/step - loss: 0.0543 - val_loss: 0.0302

Epoch 00049: val_loss improved from 0.03144 to 0.03024, saving model to models/custom-1.49-0.03
Epoch 50/100
```

```
175/175 [==============================] - 204s 1s/step - loss: 0.0547 - val_loss: 0.0321

Epoch 00050: val_loss did not improve from 0.03024
Epoch 51/100
175/175 [==============================] - 207s 1s/step - loss: 0.0547 - val_loss: 0.0306

Epoch 00051: val_loss did not improve from 0.03024
Epoch 52/100
175/175 [==============================] - 197s 1s/step - loss: 0.0549 - val_loss: 0.0309

Epoch 00052: val_loss did not improve from 0.03024
Epoch 53/100
175/175 [==============================] - 196s 1s/step - loss: 0.0543 - val_loss: 0.0334

Epoch 00053: val_loss did not improve from 0.03024
Epoch 54/100
175/175 [==============================] - 197s 1s/step - loss: 0.0551 - val_loss: 0.0343

Epoch 00054: val_loss did not improve from 0.03024
Epoch 55/100
175/175 [==============================] - 198s 1s/step - loss: 0.0534 - val_loss: 0.0341

Epoch 00055: val_loss did not improve from 0.03024
Epoch 56/100
175/175 [==============================] - 192s 1s/step - loss: 0.0523 - val_loss: 0.0345

Epoch 00056: val_loss did not improve from 0.03024
Epoch 57/100
175/175 [==============================] - 192s 1s/step - loss: 0.0539 - val_loss: 0.0328

Epoch 00057: val_loss did not improve from 0.03024
Epoch 58/100
175/175 [==============================] - 191s 1s/step - loss: 0.0528 - val_loss: 0.0316

Epoch 00058: val_loss did not improve from 0.03024
Epoch 59/100
175/175 [==============================] - 187s 1s/step - loss: 0.0526 - val_loss: 0.0324

Epoch 00059: val_loss did not improve from 0.03024
Epoch 60/100
175/175 [==============================] - 202s 1s/step - loss: 0.0515 - val_loss: 0.0327

Epoch 00060: val_loss did not improve from 0.03024
Epoch 61/100
175/175 [==============================] - 113s 648ms/step - loss: 0.0527 - val_loss: 0.0316

Epoch 00061: val_loss did not improve from 0.03024
Epoch 62/100
```

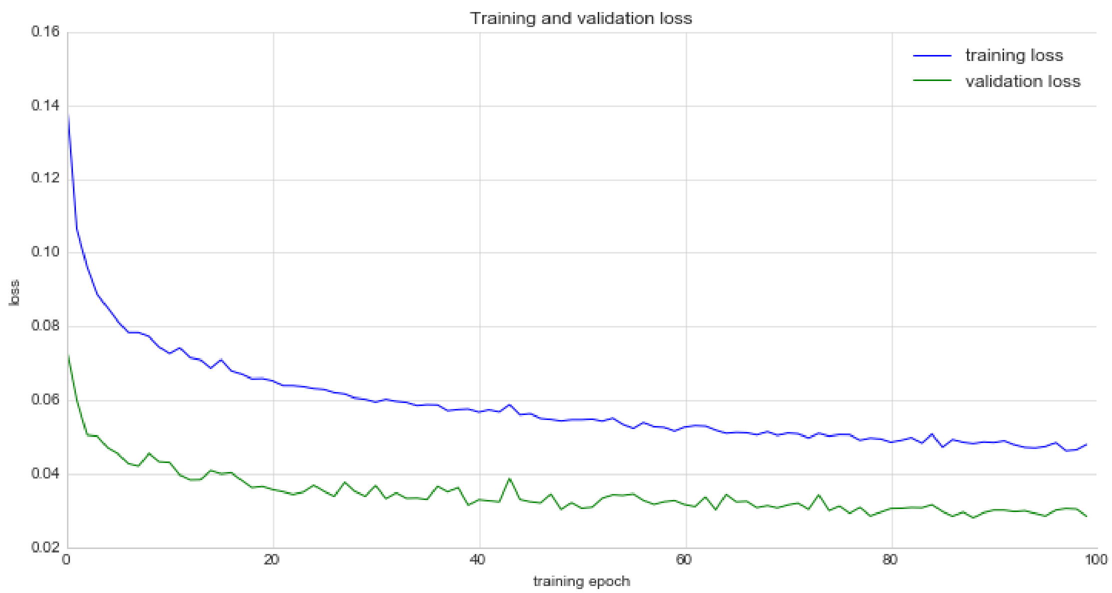
```
175/175 [==============================] - 104s 592ms/step - loss: 0.0531 - val_loss: 0.0310

Epoch 00062: val_loss did not improve from 0.03024
Epoch 63/100
175/175 [==============================] - 160s 917ms/step - loss: 0.0529 - val_loss: 0.0337

Epoch 00063: val_loss did not improve from 0.03024
Epoch 64/100
175/175 [==============================] - 159s 909ms/step - loss: 0.0518 - val_loss: 0.0301

Epoch 00064: val_loss improved from 0.03024 to 0.03014, saving model to models/custom-1.64-0.03
Epoch 65/100
175/175 [==============================] - 159s 910ms/step - loss: 0.0510 - val_loss: 0.0344

Epoch 00065: val_loss did not improve from 0.03014
Epoch 66/100
175/175 [==============================] - 161s 922ms/step - loss: 0.0512 - val_loss: 0.0324

Epoch 00066: val_loss did not improve from 0.03014
Epoch 67/100
175/175 [==============================] - 163s 931ms/step - loss: 0.0511 - val_loss: 0.0325

Epoch 00067: val_loss did not improve from 0.03014
Epoch 68/100
175/175 [==============================] - 162s 925ms/step - loss: 0.0505 - val_loss: 0.0307

Epoch 00068: val_loss did not improve from 0.03014
Epoch 69/100
175/175 [==============================] - 161s 921ms/step - loss: 0.0514 - val_loss: 0.0313

Epoch 00069: val_loss did not improve from 0.03014
Epoch 70/100
175/175 [==============================] - 161s 918ms/step - loss: 0.0504 - val_loss: 0.0307

Epoch 00070: val_loss did not improve from 0.03014
Epoch 71/100
175/175 [==============================] - 158s 902ms/step - loss: 0.0510 - val_loss: 0.0315

Epoch 00071: val_loss did not improve from 0.03014
Epoch 72/100
175/175 [==============================] - 159s 906ms/step - loss: 0.0508 - val_loss: 0.0320

Epoch 00072: val_loss did not improve from 0.03014
Epoch 73/100
175/175 [==============================] - 151s 862ms/step - loss: 0.0497 - val_loss: 0.0303

Epoch 00073: val_loss did not improve from 0.03014
Epoch 74/100
```

```
175/175 [==============================] - 123s 703ms/step - loss: 0.0510 - val_loss: 0.0343

Epoch 00074: val_loss did not improve from 0.03014
Epoch 75/100
175/175 [==============================] - 123s 705ms/step - loss: 0.0502 - val_loss: 0.0299

Epoch 00075: val_loss improved from 0.03014 to 0.02994, saving model to models/custom-1.75-0.02
Epoch 76/100
175/175 [==============================] - 123s 705ms/step - loss: 0.0506 - val_loss: 0.0312

Epoch 00076: val_loss did not improve from 0.02994
Epoch 77/100
175/175 [==============================] - 125s 716ms/step - loss: 0.0506 - val_loss: 0.0292

Epoch 00077: val_loss improved from 0.02994 to 0.02924, saving model to models/custom-1.77-0.02
Epoch 78/100
175/175 [==============================] - 114s 652ms/step - loss: 0.0491 - val_loss: 0.0308

Epoch 00078: val_loss did not improve from 0.02924
Epoch 79/100
175/175 [==============================] - 95s 545ms/step - loss: 0.0497 - val_loss: 0.0285

Epoch 00079: val_loss improved from 0.02924 to 0.02853, saving model to models/custom-1.79-0.02
Epoch 80/100
175/175 [==============================] - 96s 546ms/step - loss: 0.0494 - val_loss: 0.0295

Epoch 00080: val_loss did not improve from 0.02853
Epoch 81/100
175/175 [==============================] - 95s 545ms/step - loss: 0.0486 - val_loss: 0.0305

Epoch 00081: val_loss did not improve from 0.02853
Epoch 82/100
175/175 [==============================] - 91s 522ms/step - loss: 0.0491 - val_loss: 0.0306

Epoch 00082: val_loss did not improve from 0.02853
Epoch 83/100
175/175 [==============================] - 92s 525ms/step - loss: 0.0498 - val_loss: 0.0308

Epoch 00083: val_loss did not improve from 0.02853
Epoch 84/100
175/175 [==============================] - 92s 523ms/step - loss: 0.0483 - val_loss: 0.0307

Epoch 00084: val_loss did not improve from 0.02853
Epoch 85/100
175/175 [==============================] - 102s 582ms/step - loss: 0.0508 - val_loss: 0.0316

Epoch 00085: val_loss did not improve from 0.02853
Epoch 86/100
```

```
175/175 [==============================] - 111s 634ms/step - loss: 0.0472 - val_loss: 0.0297

Epoch 00086: val_loss did not improve from 0.02853
Epoch 87/100
175/175 [==============================] - 114s 651ms/step - loss: 0.0493 - val_loss: 0.0285

Epoch 00087: val_loss improved from 0.02853 to 0.02847, saving model to models/custom-1.87-0.0
Epoch 88/100
175/175 [==============================] - 100s 574ms/step - loss: 0.0485 - val_loss: 0.0296

Epoch 00088: val_loss did not improve from 0.02847
Epoch 89/100
175/175 [==============================] - 104s 595ms/step - loss: 0.0482 - val_loss: 0.0281

Epoch 00089: val_loss improved from 0.02847 to 0.02806, saving model to models/custom-1.89-0.0
Epoch 90/100
175/175 [==============================] - 95s 546ms/step - loss: 0.0486 - val_loss: 0.0294

Epoch 00090: val_loss did not improve from 0.02806
Epoch 91/100
175/175 [==============================] - 94s 538ms/step - loss: 0.0485 - val_loss: 0.0301

Epoch 00091: val_loss did not improve from 0.02806
Epoch 92/100
175/175 [==============================] - 92s 526ms/step - loss: 0.0489 - val_loss: 0.0301

Epoch 00092: val_loss did not improve from 0.02806
Epoch 93/100
175/175 [==============================] - 95s 544ms/step - loss: 0.0478 - val_loss: 0.0297

Epoch 00093: val_loss did not improve from 0.02806
Epoch 94/100
175/175 [==============================] - 95s 541ms/step - loss: 0.0471 - val_loss: 0.0299

Epoch 00094: val_loss did not improve from 0.02806
Epoch 95/100
175/175 [==============================] - 94s 540ms/step - loss: 0.0470 - val_loss: 0.0292

Epoch 00095: val_loss did not improve from 0.02806
Epoch 96/100
175/175 [==============================] - 94s 537ms/step - loss: 0.0474 - val_loss: 0.0285

Epoch 00096: val_loss did not improve from 0.02806
Epoch 97/100
175/175 [==============================] - 94s 537ms/step - loss: 0.0484 - val_loss: 0.0301

Epoch 00097: val_loss did not improve from 0.02806
Epoch 98/100
```

```
175/175 [==============================] - 94s 537ms/step - loss: 0.0462 - val_loss: 0.0305

Epoch 00098: val_loss did not improve from 0.02806
Epoch 99/100
175/175 [==============================] - 94s 538ms/step - loss: 0.0465 - val_loss: 0.0304

Epoch 00099: val_loss did not improve from 0.02806
Epoch 100/100
175/175 [==============================] - 94s 537ms/step - loss: 0.0479 - val_loss: 0.0285

Epoch 00100: val_loss did not improve from 0.02806
```

```
In [180]: plot_training_history(hist)
```



We find that the losses are still converging, so the network is far from overfitting. It would likely make sense to continue training, possibly using a smaller learning rate.

Either way, this model made it through the training track with ease and almost mastered the more difficult test track with only one minor incident. The recorded videos can be found at

- `video/custom-1.89-0.0281.mp4` for the training track and
- `video/custom-1-1.89-0.0281.mp4` for the test track.

For the latter, note that the car was braught back to track manually after the incident.

Here's a video of a the model driving halfway around the test track (running the training track works perfectly but looks somewhat boring at the same time).

From the driving itself, it seems that the model is steering much more often (and more aggressively at that) than before. One could possibly introduce more "zero angle" examples to help it to learn more about "smooth" driving. When running the model in the simulator problematic situations arise the model doesn't know how to resolve, such as switching lanes, driving dangerously close to the edge of the street and driving straightway into trees, straight toward and eventually off cliffs. Here, it would make sense to gather more training data specifically about those situations.

Given though that this was actually only done a handful of times during training data collection, the model does remarkably well.