self-driving-car

February 12, 2020

1 Self-Driving Introduction

Derek Riley

This notebook allows a user to train a Deep Neural Network to "learn" how to drive a simulated car. This was built by adapting a Udacity tutorial https://www.udacity.com/course/self-driving-car-engineer-nanodegree–nd013

Lets start with a few imports…

[1]:
```
import cv2
import csv
import numpy as np
import os

from keras.models import Sequential, Model
from keras.layers import Flatten, Dense, Lambda, Conv2D, Cropping2D, Dropout,␣
↪ELU
from keras.layers.pooling import MaxPooling2D
from keras.callbacks import ModelCheckpoint, Callback
from keras.regularizers import l2
from keras.optimizers import Adam

import sklearn
import matplotlib.pyplot as plt

import sys
import time
import argparse
import io
# %load_ext line_profiler
```

```
Using TensorFlow backend.
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/anaconda3/lib/python3.7/site-
```

```
packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:519: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:520: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

```
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/anaconda3/lib/python3.7/site-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

Next we define a function to create the Keras DNN. Note that minor adjustments to this model can severly break it. Please avoid the temptation to play around this until you have a solid understanding of the model and what constraints exist.

[2]:
```
def commaAiModelPrime(time_len=1):
    """
    Creates comma.ai enhanced autonomous car  model
    Replaced dropout with regularization
    Added 3 additional convolution layers
    """
    model = Sequential()
    model.add(Lambda(lambda x: (x / 255.0) - 0.5, input_shape=(160,320,3)))
    model.add(Cropping2D(cropping=((50,20), (0,0))))

    # Add three 5x5 convolution layers (output depth 64, and 64)
    model.add(Conv2D(16, (8, 8), strides=(4, 4), padding="same",␣
↪kernel_regularizer=l2(0.001)))
    model.add(ELU())
    model.add(Conv2D(32, (5, 5), strides=(2, 2), padding="same",␣
↪kernel_regularizer=l2(0.001)))
    model.add(ELU())
    model.add(Conv2D(48, (5, 5), strides=(2, 2), padding="same",␣
↪kernel_regularizer=l2(0.001)))
    model.add(ELU())

    # Add two 3x3 convolution layers (output depth 64, and 64)
    model.add(Conv2D(64, (3, 3), padding='valid', kernel_regularizer=l2(0.001)))
    model.add(ELU())
    model.add(Conv2D(64, (3, 3), padding='valid', kernel_regularizer=l2(0.001)))
    model.add(ELU())

    model.add(Flatten())

    # model.add(Dropout(.2))
    model.add(Dense(100, kernel_regularizer=l2(0.001)))
    model.add(ELU())

    # model.add(Dropout(0.50))
    model.add(Dense(50, kernel_regularizer=l2(0.001)))
    model.add(ELU())
```

```
    # model.add(Dropout(0.50))
    model.add(Dense(10, kernel_regularizer=l2(0.001)))
    model.add(ELU())

    model.add(Dense(1))

    model.compile(optimizer=Adam(lr=1e-4), loss='mse')

    return model
```

The following functions load the driving data and images.

[3]:
```
def getDrivingLogs(path, skipHeader=False):
    """
    Returns the lines from a driving log with base directory `dataPath`.
    If the file include headers, pass `skipHeader=True`.
    """
    lines = []
    with open(path + '/driving_log.csv') as csvFile:
        reader = csv.reader(csvFile)
        if skipHeader:
            next(reader, None)
        for line in reader:
            lines.append(line)
    return lines


def getImages(path):
    """
    Get all training images on the path `dataPath`.
    Returns `([centerPaths], [leftPath], [rightPath], [measurement])`
    """
    directories = [x[0] for x in os.walk(path)]
    dataDirectories = list(filter(lambda directory: os.path.isfile(
        directory + '/driving_log.csv'), directories))
    print(dataDirectories)
    centerTotal = []
    leftTotal = []
    rightTotal = []
    measurementTotal = []
    for directory in dataDirectories:
        lines = getDrivingLogs(directory, skipHeader=True)
        center = []
        left = []
        right = []
        measurements = []
        for line in lines:
```

```
            measurements.append(float(line[3]))
            center.append(directory + '/' + line[0].strip())
            left.append(directory + '/' + line[1].strip())
            right.append(directory + '/' + line[2].strip())
        centerTotal.extend(center)
        leftTotal.extend(left)
        rightTotal.extend(right)
        measurementTotal.extend(measurements)

    return (centerTotal, leftTotal, rightTotal, measurementTotal)
```

The following function combines images to prepare them to feed into the network.

[4]:
```
def combineCenterLeftRightImages(center, left, right, measurement, correction):
    """
    Combine the image paths from `center`, `left` and `right` using the␣
 ↪correction factor `correction`
    Returns ([imagePaths], [measurements])
    """
    imagePaths = []
    imagePaths.extend(center)
    imagePaths.extend(left)
    imagePaths.extend(right)
    measurements = []
    measurements.extend(measurement)
    measurements.extend([x + correction for x in measurement])
    measurements.extend([x - correction for x in measurement])
    return (imagePaths, measurements)
```

This generator provides data to train and validate the neural network.

[5]:
```
def generator(samples, batch_size=32, flip = True):
    """
    Generate the required images and measurments for training/
    `samples` is a list of pairs (`imagePath`, `measurement`).
    """
    num_samples = len(samples)
    while 1: # Loop forever so the generator never terminates
        samples = sklearn.utils.shuffle(samples)
        for offset in range(0, num_samples, batch_size):
            batch_samples = samples[offset:offset+batch_size]

            images = []
            angles = []
            for imagePath, measurement in batch_samples:
                originalImage = cv2.imread(imagePath)
                image = cv2.cvtColor(originalImage, cv2.COLOR_BGR2RGB)
```

```
                images.append(image)
                angles.append(measurement)
                # Flipping
                if(flip):
                    images.append(cv2.flip(image,1))
                    angles.append(measurement*-1.0)

            # trim image to only see section with road
            inputs = np.array(images)
            outputs = np.array(angles)
            yield sklearn.utils.shuffle(inputs, outputs)
```

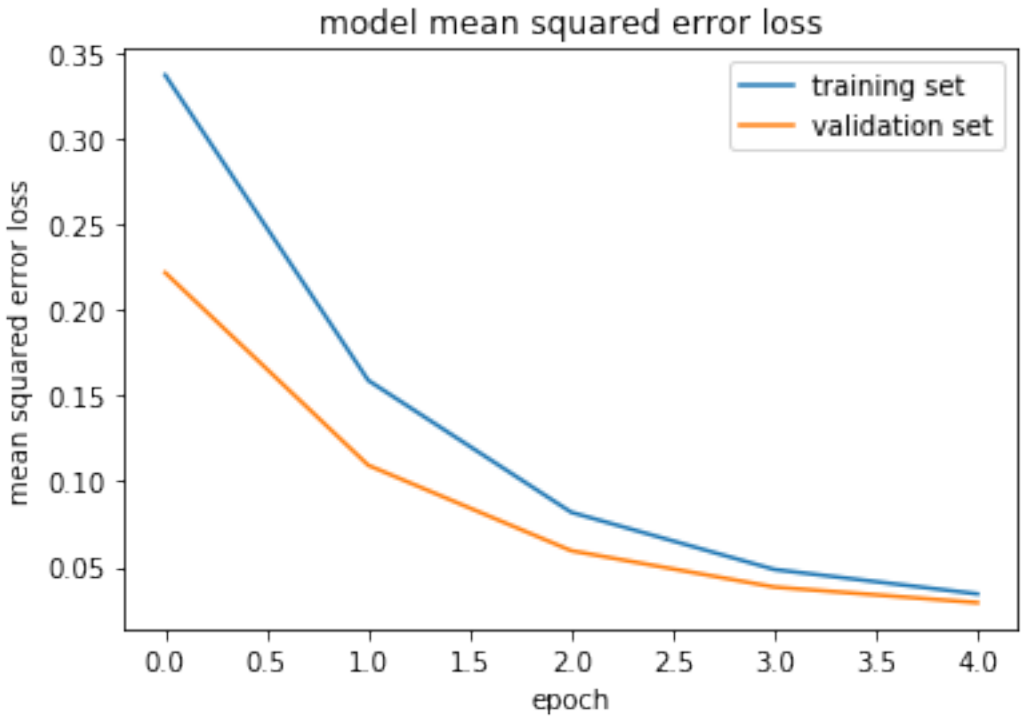
Below is the "main" functionality of this training. You will need to adjust the data paths, and you will adjust the batch size and number of epochs.

[6]:
```
my_home = "/home/harleys/CS-2300/lab8/SelfDrivingSim"
my_data = "/data/cs2300/L8"
my_batch_size = 32
my_epochs = 5

# Reading images locations.
centerPaths, leftPaths, rightPaths, measurements = getImages(my_data)
imagePaths, measurements = combineCenterLeftRightImages(
    centerPaths, leftPaths, rightPaths, measurements, 0.2)
print('Total Images: {}'.format(len(imagePaths)))

# Splitting samples and creating generators.
from sklearn.model_selection import train_test_split

samples = list(zip(imagePaths, measurements))
train_samples, validation_samples = train_test_split(samples, test_size=0.2)

print('Train samples: {}'.format(len(train_samples)))
print('Validation samples: {}'.format(len(validation_samples)))

train_generator = generator(train_samples, batch_size=my_batch_size)
validation_generator = generator(validation_samples, batch_size=my_batch_size)

# Model creation
model = commaAiModelPrime()

# Train the model
history_object = model.fit_generator(train_generator, steps_per_epoch= \
    len(train_samples)/my_batch_size, validation_data=validation_generator, \
    validation_steps=len(validation_samples)/my_batch_size, epochs=my_epochs,␣
 ↪verbose=1)
```

```
model.save(my_home + '/model_commaAiModelPrime_e5.h5')

print(history_object.history.keys())
print('Loss')
print(history_object.history['loss'])
print('Validation Loss')
print(history_object.history['val_loss'])

plt.plot(history_object.history['loss'])
plt.plot(history_object.history['val_loss'])
plt.title('model mean squared error loss')
plt.ylabel('mean squared error loss')
plt.xlabel('epoch')
plt.legend(['training set', 'validation set'], loc='upper right')
plt.show()
```

```
['/data/cs2300/L8']
Total Images: 24108
Train samples: 19286
Validation samples: 4822
WARNING:tensorflow:From /usr/local/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/anaconda3/lib/python3.7/site-
packages/keras/optimizers.py:790: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

Epoch 1/5
603/602 [==============================] - 110s 182ms/step - loss: 0.3376 -
val_loss: 0.2221
```

```
Epoch 2/5
603/602 [==============================] - 97s 161ms/step - loss: 0.1590 -
val_loss: 0.1094
Epoch 3/5
603/602 [==============================] - 97s 161ms/step - loss: 0.0819 -
val_loss: 0.0595
Epoch 4/5
603/602 [==============================] - 102s 170ms/step - loss: 0.0485 -
val_loss: 0.0382
Epoch 5/5
603/602 [==============================] - 107s 177ms/step - loss: 0.0342 -
val_loss: 0.0291
dict_keys(['val_loss', 'loss'])
Loss
[0.33763193343822534, 0.15906421153621778, 0.0819208895833762,
0.048486342197580606, 0.03422273350870858]
Validation Loss
[0.222065171862677, 0.10943106655533608, 0.05946092056794218,
0.03822227768609258, 0.02909441062835231]
```



[7]:
```
def create_model():
    my_home = "/home/harleys/CS-2300/lab8/SelfDrivingSim"
    my_data = "/home/harleys/CS-2300/lab8"
    my_batch_size = 32
```

```
    my_epochs = 20

    # Reading images locations.
    centerPaths, leftPaths, rightPaths, measurements = getImages(my_data)
    imagePaths, measurements = combineCenterLeftRightImages(
        centerPaths, leftPaths, rightPaths, measurements, 0.2)
    print('Total Images: {}'.format(len(imagePaths)))

    # Splitting samples and creating generators.
    from sklearn.model_selection import train_test_split

    samples = list(zip(imagePaths, measurements))
    train_samples, validation_samples = train_test_split(samples, test_size=0.2)

    print('Train samples: {}'.format(len(train_samples)))
    print('Validation samples: {}'.format(len(validation_samples)))

    train_generator = generator(train_samples, batch_size=my_batch_size)
    validation_generator = generator(validation_samples,␣
↪batch_size=my_batch_size)

    # Model creation
    model = commaAiModelPrime()

    # Train the model
    history_object = model.fit_generator(train_generator, steps_per_epoch= \
        len(train_samples)/my_batch_size, validation_data=validation_generator,␣
↪\
        validation_steps=len(validation_samples)/my_batch_size,␣
↪epochs=my_epochs, verbose=1)

    model.save(my_home + '/final_model.h5')

    print(history_object.history.keys())
    print('Loss')
    print(history_object.history['loss'])
    print('Validation Loss')
    print(history_object.history['val_loss'])

    plt.plot(history_object.history['loss'])
    plt.plot(history_object.history['val_loss'])
    plt.title('model mean squared error loss')
    plt.ylabel('mean squared error loss')
    plt.xlabel('epoch')
    plt.legend(['training set', 'validation set'], loc='upper right')
    plt.show()
```

[8]:
```
# %lprun -f create_model create_model()
```