UDACITY

‹ Back to Self-Driving Car Engineer

Use Deep Learning to Clone Driving Behavior

REVIEW

CODE REVIEW 3

HISTORY

▾ model.py 3

```
#Imports

from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
import matplotlib.pyplot as plt
import tensorflow as tf
from scipy import stats
from time import time
import numpy as np
import datetime
import sklearn
import random
import keras
import math
import csv
import cv2
import sys
import os


#import keras modules
from keras.layers.core import Dense, Activation, Flatten, Dropout, Lambda
from keras.layers.convolutional import Convolution2D, ZeroPadding2D
from keras.utils.layer_utils import layer_from_config
from keras.layers.pooling import MaxPooling2D
from keras.models import  Model, Sequential
from keras.callbacks import ModelCheckpoint
from keras.optimizers import Adam , SGD
from keras.layers import Cropping2D,Input
from keras.regularizers import l2
from keras.utils import np_utils
import keras


#############################################################################

def generator(samples, batch_size=32,data_path='IMG/',corrl=0,corrh=0):
    angleref=0.5
    num_samples = len(samples)
    while 1: # Loop forever so the generator never terminates
        shuffle(samples)
        for offset in range(0, num_samples, batch_size):
            batch_samples = samples[offset:offset+batch_size]

            images = []
            angles = []
            for batch_sample in batch_samples:
                name = data_path+batch_sample[0].split('/')[-1]
                center_image = cv2.imread(name)
                name = data_path+batch_sample[1].split('/')[-1]
                left_image = cv2.imread(name)
                name = data_path+batch_sample[2].split('/')[-1]
                right_image = cv2.imread(name)

                center_angle = float(batch_sample[3])
                images.append(center_image)
                images.append(left_image)
                images.append(right_image)
                if  float(center_angle)>=float(angleref) or float(center_angle)<
                    angles.append(center_angle)
                    angles.append(center_angle+corrl)
                    angles.append(center_angle-corrl)
                else:
                    angles.append(center_angle)
                    angles.append(center_angle+corrh)
                    angles.append(center_angle-corrh)

                augmented_images, augmented_angles=[],[]

            for image,angle in zip(images,angles):
                augmented_images.append(image)
                augmented_angles.append(angle)
                augmented_images.append(cv2.flip(image,1))
                augmented_angles.append(angle*-1)


            # trim image to only see section with road
            X_train = np.array(augmented_images)
            y_train = np.array(augmented_angles)
            yield sklearn.utils.shuffle(X_train, y_train)

################################################################################

#define model architecture
model = Sequential()
```

```

# NVIDIA model
#define model architecture
model.add(Lambda(lambda x: x/255.0 - 0.5,input_shape=(160,320,3),output_shape=(1
model.add(Cropping2D(cropping=((50,30), (0,0)), input_shape=(160,320,3)))
model.add(Convolution2D(24,5,5, subsample=(2,2),activation='relu',border_mode =
model.add(Convolution2D(36,5,5, subsample=(2,2),activation='relu',border_mode =
model.add(Convolution2D(48,5,5, subsample=(2,2),activation='relu',border_mode =
model.add(Convolution2D(64,3,3, activation='relu',name='conv1_4'))
model.add(Convolution2D(64,3,3, activation='relu',name='conv1_5'))

model.add(Flatten())#input_shape=model.output_shape[1:]
model.add(Dropout(0.5, name='drop1'))
```

AWESOME

Good job using dropout layer in the network to reduce overfitting.

```
model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(10, activation='relu'))
```

AWESOME

Well done using ReLU activation to introduce non-linearity into the network.

```
#Output
model.add(Dense(1))#input_shape=model_classification.output_shape[1:]

model.summary()

################################################################################

def train_model(train_samples,validation_samples,data_path,model_name,batch_size

    # compile model
    #model.compile(optimizer='adam', loss='mse',metrics=['accuracy'])
    model.compile(optimizer=Adam(lr=learningrate), loss='mse')
    #model.compile(loss='categorical_crossentropy',optimizer='adadelta',metrics=
    #model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['ac
    #model.compile(loss='binary_crossentropy', optimizer=SGD(lr=learningrate, mo

    #using the generator function
    train_generator = generator(train_samples, batch_size=batch_size,data_path=d
```

AWESOME

Nice work choosing appropriate training data to keep the car on the track.

```
    validation_generator = generator(validation_samples, batch_size=batch_size,d


    now = datetime.datetime.now
    #define train task
    t = now()
    history = model.fit_generator(train_generator,
                                  samples_per_epoch=len(train_samples)*6,
                                  validation_data=validation_generator,
                                  nb_val_samples=len(validation_samples)*6,
                                  nb_epoch=nb_epoch,
                                  verbose=1)

    print('Training time: %s' % (now() - t))

    #save the model
    model.save(model_name+'.h5')
    model.save_weights(model_name+'_weights.h5')
    #with open('model.json', 'w') as outfile:outfile.write(model.to_json())
    print ("training finish")

    #Visualize
    ### print the keys contained in the history object
    print(history.history.keys())
    ### plot the training and validation loss for each epoch
    plt.plot(history.history['loss'])
    plt.plot(history.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()

################################################################################

def tf_learning_case(tlcase=3):

    if (tlcase==1):
    #freeze Convolutional and Classification layers Case 1: Small Data Set, Simi
        for l in model.layers[:-1]:
            l.trainable = False
        model.pop()
        model.add(Dense(1,init='uniform'))

    elif (tlcase==2):
    #freeze Convolutional layers Case 2: Small Data Set, Different Data
        for l in model.layers[:-7]:
            l.trainable = False
        model.pop()
        model.pop()
        model.pop()
        model.pop()
        model.add(Dense(1,init='uniform'))

    elif (tlcase==3):
    #Output layer random initial Case 3: Large Data Set, Similar Data
        model.pop()
        model.add(Dense(1,init='uniform'))
################################################################################

#import data for first data set
samples = []
dataset="uda"
imgpath=""

if dataset=="uda":

    imgpath="IMG_Udacity/"
    with open('driving_log_Udacity.csv') as csvfile:
        reader = csv.reader(csvfile)
        for line in reader:
            samples.append(line)

elif dataset=="ps3" :

    imgpath="IMG_PS3/"
    with open('driving_log_PS3.csv') as csvfile:
        reader = csv.reader(csvfile)
        for line in reader:
            samples.append(line)

elif dataset=="ps3inv":

    imgpath="IMG_PS3_INV/"
    with open('driving_log_PS3_INV.csv') as csvfile:
        reader = csv.reader(csvfile)
        for line in reader:
            samples.append(line)


train_samples, validation_samples = train_test_split(samples, test_size=0.2)
train_model(train_samples,validation_samples,data_path=imgpath,model_name='model

###########################################################################
#import data for transfer  learning  , here I use fine tuning  that corresponds

model.load_weights('model_weights.h5')

#import data for transfer learning train
samples = []
with open('driving_log_PS3.csv') as csvfile:
    reader = csv.reader(csvfile)
    for line in reader:
        samples.append(line)

train_samples, validation_samples = train_test_split(samples, test_size=0.2)

tf_learning_case(3)

train_model(train_samples,validation_samples,data_path='IMG_PS3/',model_name='mo
```

▸ drive.py

▸ writeup_report.html

▸ writeup_report.md

RETURN TO PATH

Rate this review