Deep Learning in Practice

Anis Koubaa

The Vehicle Type Classification Project

Evaluation of the classifier

We first need to load the required libraries

```
%load_ext autoreload
%autoreload 2

# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import AveragePooling2D, GlobalAveragePooling2D, Batch
#from tensorflow.keras.applications import Xception
#from tensorflow.keras.applications import resnet50
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import cv2
import os
import sys
import tensorflow as tf
import h5py
import numpy as np
import sys
```

```
#set the model type that you will train in this notebook
```

```
  ascii_dev_labels = np.array(hf["devLabels"]).astype("S65")
  devY=np.array(hf["devY"]).astype("int")

  testX= np.array(hf["testX"]).astype("f8")
  ascii_test_labels = np.array(hf["testLabels"]).astype("S65")
  testY=np.array(hf["testY"]).astype("int")

  trainLabels = np.array([n.decode('unicode_escape') for n in ascii_train_labels])
  devLabels = np.array([n.decode('unicode_escape') for n in ascii_dev_labels])
  testLabels = np.array([n.decode('unicode_escape') for n in ascii_test_labels])

  print("trainX.shape: ",trainX.shape)
  print("trainY.shape: ",trainY.shape)
  print("trainLabels.shape: ",trainLabels.shape)
  print("devX.shape: ",devX.shape)
  print("devY.shape: ",devY.shape)
  print("devLabels.shape: ",devLabels.shape)
  print("testX.shape: ",testX.shape)
  print("testY.shape: ",testY.shape)
  print("testLabels.shape: ",testLabels.shape)

  return trainX, trainY, trainLabels, devX,devY,devLabels,testX,testY,testLabels


def load_dev_test_dataset_from_hdf5_file(hdf_file_path):
  hf = h5py.File(hdf_file_path, "r")

  devX= np.array(hf["devX"]).astype("f8")
  ascii_dev_labels = np.array(hf["devLabels"]).astype("S65")
  devY=np.array(hf["devY"]).astype("int")

  testX= np.array(hf["testX"]).astype("f8")
  ascii_test_labels = np.array(hf["testLabels"]).astype("S65")
  testY=np.array(hf["testY"]).astype("int")


  devLabels = np.array([n.decode('unicode_escape') for n in ascii_dev_labels])
  testLabels = np.array([n.decode('unicode_escape') for n in ascii_test_labels])


  print("devX.shape: ",devX.shape)
  print("devY.shape: ",devY.shape)
  print("devLabels.shape: ",devLabels.shape)
  print("testX.shape: ",testX.shape)
  print("testY.shape: ",testY.shape)
  print("testLabels.shape: ",testLabels.shape)

  return devX,devY,devLabels,testX,testY,testLabels


import datetime
t0 = datetime.datetime.now()
devX,devY,devLabels,testX,testY,testLabels=load_dev_test_dataset_from_hdf5_file(HDF
t1 = datetime.datetime.now()
print('time to load data: ', (t1-t0))
```

```
TYPE='type'
model_type='mobilenetv2'
#set your user name
user='anis'
iteration='1'

#NOTE: Make sure to set the correct project path with respect to your Google Drive

PROJECT_PATH='/content/drive/My Drive/udemy-deep-learning-in-practice/03-transfer-l
print('PROJECT_PATH: ',PROJECT_PATH)

HDF5_DATASET_PATH=PROJECT_PATH+'datasets/vehicle-type-dataset-SIZE224-train-dev-tes
print('HDF5_DATASET_PATH: ', HDF5_DATASET_PATH)

ACCURACY_LOSS_OUPUT_FILE=PROJECT_PATH+'log/'+model_type+'/'+model_type+'-by-'+TYPE+
print('ACCURACY_LOSS_OUPUT_FILE: ', ACCURACY_LOSS_OUPUT_FILE)

TRAINED_MODEL=PROJECT_PATH+'trained-models/'+model_type+'/'+'vehicle-classification
print('TRAINED_MODEL: ',TRAINED_MODEL)

CHECKPOINT_PATH = PROJECT_PATH+'checkpoints/'+model_type+'/'+'by-'+TYPE+'-'+model_t
print('CHECKPOINT_PATH: ',CHECKPOINT_PATH)

LOGFILE_PATH=PROJECT_PATH+'log/'+model_type+'/'+model_type+'-by-'+TYPE+'-training-l
print('LOGFILE_PATH: ',LOGFILE_PATH)
```

```
PROJECT_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-trar
HDF5_DATASET_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03
ACCURACY_LOSS_OUPUT_FILE:  /content/drive/My Drive/udemy-deep-learning-in-prac
TRAINED_MODEL:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-tra
CHECKPOINT_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-t
LOGFILE_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-trar
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
sys.path.append(PROJECT_PATH)
import anis_koubaa_udemy_computer_vision_lib
from anis_koubaa_udemy_computer_vision_lib import *
```

Load the Dataset

```
def load_dataset_from_hdf5_file(hdf_file_path):
  hf = h5py.File(hdf_file_path, "r")
  trainX= np.array(hf["trainX"]).astype("f8")
  ascii_train_labels = np.array(hf["trainLabels"]).astype("S65")
  trainY=np.array(hf["trainY"]).astype("int")

  devX= np.array(hf["devX"]).astype("f8")
```

```
devX.shape:  (75, 224, 224, 3)
devY.shape:  (75, 7)
devLabels.shape:  (75,)
testX.shape:  (76, 224, 224, 3)
testY.shape:  (76, 7)
testLabels.shape:  (76,)
time to load data:  0:00:03.916500
```

```
IMAGE_SIZE=testX.shape[1]
print(IMAGE_SIZE)
```

```
224
```

Dataset Visualization

```
#anis_koubaa_udemy_computer_vision_lib.plot_sample_from_dataset(trainX, trainLabels
```

```
anis_koubaa_udemy_computer_vision_lib.plot_sample_from_dataset(devX, devLabels,rows
```


```
anis_koubaa_udemy_computer_vision_lib.plot_sample_from_dataset(testX, testLabels,ro
```


```
# Convert to TF Serving
```

```
#print('Loading the best model...')
model = tf.keras.models.load_model(CHECKPOINT_PATH)


#tf.compat.v1.keras.experimental.export_saved_model(model, PROJECT_PATH+'tensorflow
```

Make Predictions

```
print('Loading the best model...', TRAINED_MODEL)
t0 = datetime.datetime.now()
test_model = tf.keras.models.load_model(TRAINED_MODEL)
t1 = datetime.datetime.now()
print('time to load the model: ', (t1-t0))
```

```
Loading the best model... /content/drive/My Drive/udemy-deep-learning-in-pract
time to load the model:  0:00:01.978342
```

```
# make predictions on the testing set
print("[INFO] evaluating network on the dev dataset...")
test_model.evaluate(devX, devY,verbose=0)
```

```
[INFO] evaluating network on the dev dataset...
[1.1841479539871216, 1.0]
```

```
print("[INFO] evaluating network on the test dataset...")
test_model.evaluate(testX, testY,verbose=0)
```

```
[INFO] evaluating network on the test dataset...
[1.2156026363372803, 0.9736841917037964]
```

```
class_dict,number_of_classes=get_cars_classes_dict(testY,testLabels)
class_dict
```

```
{0: 'car-bus-alltypes',
 1: 'car-sedan-alltypes',
 2: 'car-suv-alltypes',
 3: 'motocycle-bicycle-kids',
 4: 'motocycle-bicycle-racing',
 5: 'motocycle-motorbike-chopper',
 6: 'motocycle-motorbike-sport'}
```

```
class_dict[6]
```

```
'motocycle-motorbike-sport'
```

```
len(class_dict)
```

```
7
```

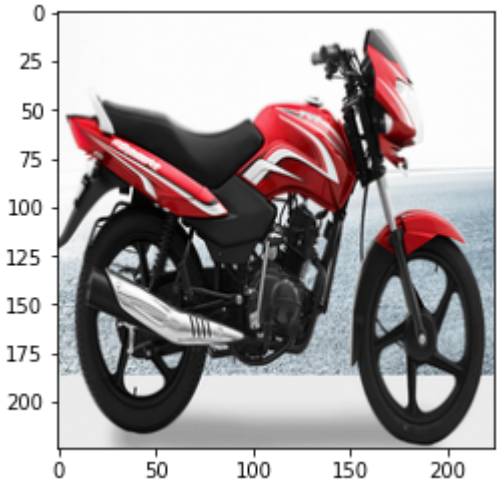
```
len(class_dict)
```

```
7
```

Testing on one image

```
index=30
test_image = testX[index]
print(testY[index])
print(class_dict[np.argmax(testY[index])])
anis_koubaa_udemy_computer_vision_lib.display_image(testX, testLabels, index)
```

```
[0 0 0 0 0 0 1]
motocycle-motorbike-sport
Label = motocycle-motorbike-sport
image shape:  (224, 224, 3)
```



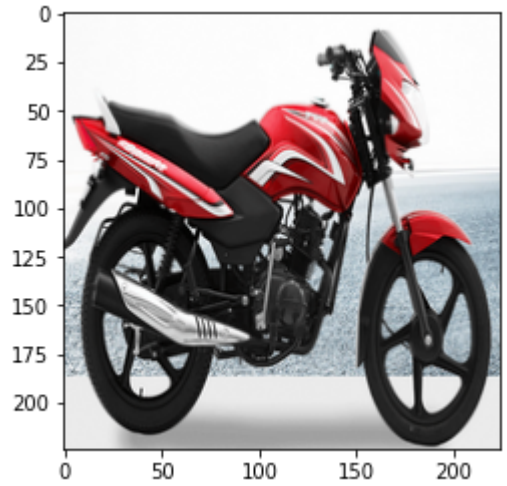

```
import datetime
```

```
plt.imshow(test_image)
reshaped_image=anis_koubaa_udemy_computer_vision_lib.reshape_image_for_neural_netwo
```

```
t0 = datetime.datetime.now()
prediction_array = test_model.predict(reshaped_image)
t1 = datetime.datetime.now()
print('----------------------------------------')
print ("prediction time: ", t1-t0)
print ("true label: ", testY[index])
print ("predicted label: ", np.argmax(prediction_array[0]))
print ("Car Brand: ",class_dict[np.argmax(testY[index])])
```

```
print ("Car Brand Predicted: ",class_dict[np.argmax(prediction_array[0])])
print ("Confidence: ",prediction_array[0][np.argmax(prediction_array[0])])
print('----------------------------------------')
```

```
flatten the image
image.shape (150528, 1)
reshape the image to be similar to the input feature vector
image.shape (1, 224, 224, 3)
----------------------------------------
prediction time:  0:00:00.667347
true label:  [0 0 0 0 0 0 1]
predicted label:  6
Car Brand:  motocycle-motorbike-sport
Car Brand Predicted:  motocycle-motorbike-sport
Confidence:  0.99122167
----------------------------------------
```




```
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predicted_label = np.argmax(testY[index])
print("true label: ", np.argmax(prediction_array[0]))
print("predicted label: ", predicted_label)
```

```
true label:  6
predicted label:  6
```

```
reshaped_test_image = anis_koubaa_udemy_computer_vision_lib.reshape_image_for_neura
p = test_model.predict(reshaped_test_image)
print(p)
class_dict[np.argmax(p[0])]
```

```
flatten the image
image.shape (150528, 1)
reshape the image to be similar to the input feature vector
image.shape (1, 224, 224, 3)
[[2.7564276e-04 4.7135452e-04 9.2403946e-04 8.2937750e-04 8.1273646e-04
  5.4651937e-03 9.9122167e-01]]
'motocycle-motorbike-sport'
```

Testing with the whole dataset

```
prediction_array_all_test_dataset = test_model.predict(testX, verbose=1)
predicted_labels_all_test_dataset = np.argmax(prediction_array_all_test_dataset, ax

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predicted_labels_all_test_dataset
```

```
1/3 [=========>.....................] - ETA: 0sWARNING:tensorflow:Callbacks met
3/3 [==============================] - 0s 19ms/step
                              precision    recall  f1-score   support

                car-bus-alltypes       1.00      1.00      1.00        13
               car-sedan-alltypes       0.92      1.00      0.96        12
                 car-suv-alltypes       1.00      0.89      0.94         9
           motocycle-bicycle-kids       0.93      1.00      0.96        13
      motocycle-bicycle-racing       1.00      0.90      0.95        10
    motocycle-motorbike-chopper       1.00      1.00      1.00         9
      motocycle-motorbike-sport       1.00      1.00      1.00        10

                       accuracy                           0.97        76
                      macro avg       0.98      0.97      0.97        76
                   weighted avg       0.98      0.97      0.97        76
```

```
# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
model_confusion_matrix = confusion_matrix(testY.argmax(axis=1), predicted_labels_al
# true_positive = model_confusion_matrix[0, 0]
# true_negative= model_confusion_matrix[1, 1]
# false_positive=model_confusion_matrix[0, 1]
# false_negative=model_confusion_matrix[1, 0]

# total = sum(sum(model_confusion_matrix))

# acc = (true_positive + true_negative) / total
# sensitivity = true_positive / (true_positive + false_negative)
# precision = true_positive / (true_positive + false_positive)
# recall = sensitivity
# specificity = true_negative / (false_positive + true_negative)
# precision = true_positive / (true_positive + false_positive)


# show the confusion matrix, accuracy, sensitivity, and specificity
print("model_confusion_matrix: \n",model_confusion_matrix)
#print("acc: {:.4f}".format(acc))
#print("sensitivity, recall: {:.4f}".format(sensitivity))
#print("specificity: {:.4f}".format(specificity))
```

```
```

```
model_confusion_matrix:
 [[13  0  0  0  0  0  0]
 [ 0 12  0  0  0  0  0]
 [ 0  1  8  0  0  0  0]
 [ 0  0  0 13  0  0  0]
 [ 0  0  0  1  9  0  0]
```

```
#anis_koubaa_udemy_computer_vision_lib.plot_loss_accuracy(H,EPOCHS=25, output_file=
```

Double-click (or enter) to edit

```
#classes_names=['covid', 'normal']
#print(testX.shape)
#print(test_binary_labels.shape)
#anis_koubaa_udemy_computer_vision_lib.plot_car_sample_predictions(testX, predictio
anis_koubaa_udemy_computer_vision_lib.plot_car_sample_predictions_v2(testX, testY,
```

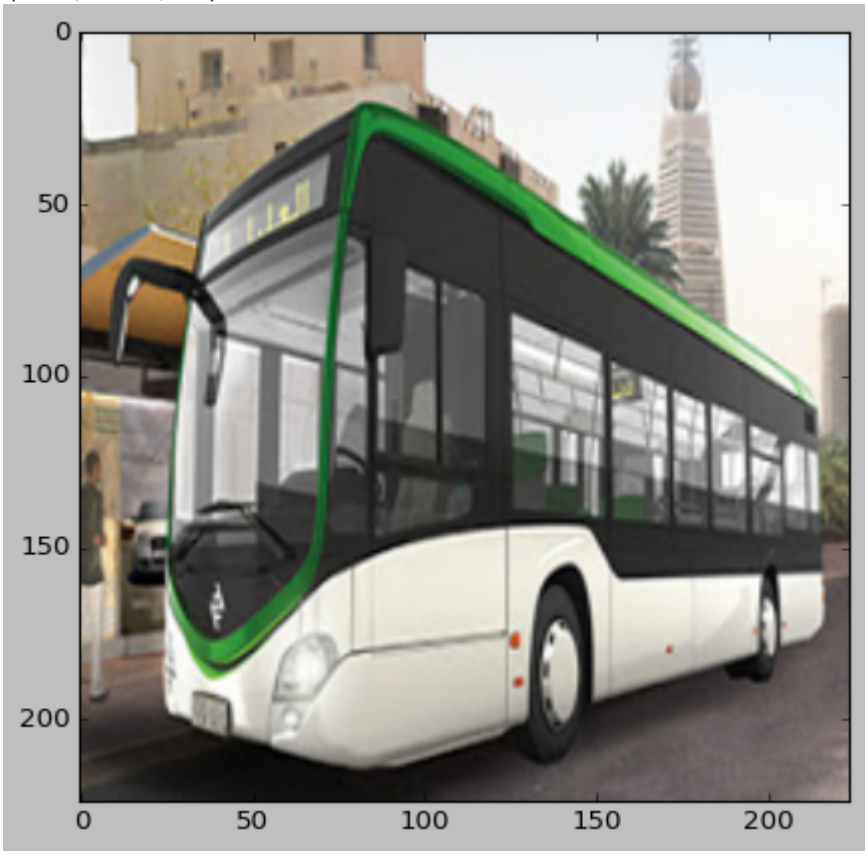

```
from PIL import Image
test_image = Image.open(PROJECT_PATH+'test-images/bus.jpg')

IMAGE_SIZE=224

test_image = test_image.resize((IMAGE_SIZE, IMAGE_SIZE))

print(test_image.format)
print(test_image.size)
print(test_image.mode)
test_image = np.array(test_image)
print (test_image.shape)
if (test_image.shape[2]>3): #sometime the image comes with RGBA with 4 channels
    test_image = cv2.cvtColor(test_image, cv2.COLOR_BGRA2BGR)
print (test_image.shape)
x=plt.imshow(test_image)
```

```
None
(224, 224)
RGB
(224, 224, 3)
(224, 224, 3)
```




```
reshaped_test_image = anis_koubaa_udemy_computer_vision_lib.reshape_image_for_neura
prediction_my_image = test_model.predict(reshaped_test_image)
print(prediction_my_image)
class_dict[np.argmax(prediction_my_image)]
```

```
flatten the image
image.shape (150528, 1)
reshape the image to be similar to the input feature vector
image.shape (1, 224, 224, 3)
[[9.9798238e-01 2.2007358e-04 3.4697400e-04 2.9347328e-04 5.6409533e-04
  3.5445308e-04 2.3848382e-04]]
'car-bus-alltypes'
```

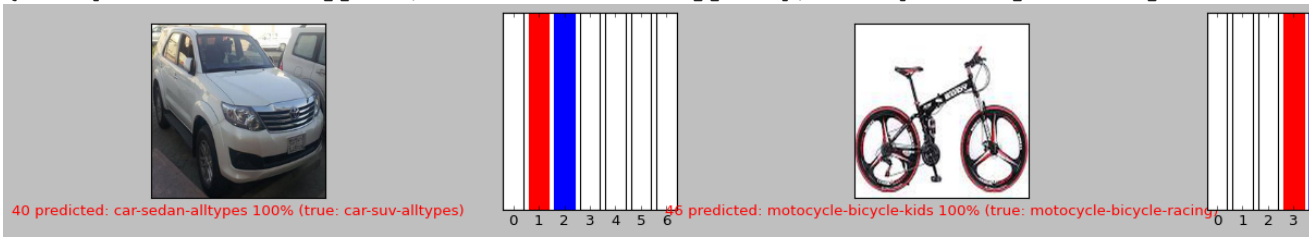
```
def top_five(prediction_my_image, class_dict):
    sorted_array=np.argsort(prediction_my_image)[0][-5:]
    sorted_array=np.flip(sorted_array)
    sorted_array
    for s in sorted_array:
    #print(class_dict[s])
    #print('id:',s,', brand:'+class_dict[s]+', confidence:',prediction_my_image[0,s
        print('id: %3d, brand:%-10s, confidence: %6.3f'%(s,class_dict[s],prediction

top_five(prediction_my_image, class_dict)
```

```
id:   0, brand:car-bus-alltypes, confidence:  0.998
id:   4, brand:motocycle-bicycle-racing, confidence:  0.001
id:   5, brand:motocycle-motorbike-chopper, confidence:  0.000
id:   2, brand:car-suv-alltypes, confidence:  0.000
id:   3, brand:motocycle-bicycle-kids, confidence:  0.000
```

```
t0=datetime.datetime.now()
ms_test=anis_koubaa_udemy_computer_vision_lib.plot_misclassifications(testX, testY,
t1=datetime.datetime.now()
print('time to find misclassifications: ',(t1-t0))
```

```
number of misclassications:  2
number of images: 75
rate of misclassications:  2.6666666666666665 %
{40: ['car-suv-alltypes', 'car-sedan-alltypes'], 46: ['motocycle-bicycle-racin
```




```
time to find misclassifications:  0:00:00.317932
```

```
prediction_array_all_dev_dataset = test_model.predict(devX, verbose=1)
```

```
1/3 [=========>.....................] - ETA: 0sWARNING:tensorflow:Callbacks met
3/3 [==============================] - 0s 19ms/step
```

```
t0=datetime.datetime.now()
ms_dev=anis_koubaa_udemy_computer_vision_lib.plot_misclassifications(devX, devY, de
t1=datetime.datetime.now()
print('time to find misclassifications: ',(t1-t0))
```

```
number of misclassications:  0
number of images: 75
rate of misclassications:  0.0 %
{}
<Figure size 3200x3200 with 0 Axes>
time to find misclassifications:  0:00:00.018239
```