Transfer learning in image classification

In this notebook we will use transfer learning and take pre-trained model from google's Tensorflow Hub and re-train that on flowers dataset. Using pre-trained model saves lot of time and computational budget for new classification problem at hand

Entrée [1]:

```
! pip install tensorflow_hub
```

```
Collecting tensorflow_hub
  Downloading tensorflow_hub-0.12.0-py2.py3-none-any.whl (108 kB)
Requirement already satisfied: protobuf>=3.8.0 in d:\ananconda\lib\site-pack
ages (from tensorflow_hub) (3.19.4)
Requirement already satisfied: numpy>=1.12.0 in d:\ananconda\lib\site-packag
es (from tensorflow_hub) (1.20.3)
Installing collected packages: tensorflow-hub
Successfully installed tensorflow-hub-0.12.0
```

Entrée [2]:

```
import numpy as np
import cv2

import PIL.Image as Image
import os

import matplotlib.pylab as plt

import tensorflow as tf
import tensorflow_hub as hub

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

Make predictions using ready made model (without any training)

Entrée [5]:

```
IMAGE_SHAPE = (224, 224)

classifier = tf.keras.Sequential([
    hub.KerasLayer("https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4", in
])
```

Entrée [9]:

```
gold_fish = Image.open("goldfish.jpg").resize(IMAGE_SHAPE)
gold_fish
```

Out[9]:

Entrée [10]:

```
gold_fish = np.array(gold_fish)/255.0
gold_fish.shape
```

Out[10]:

```
(224, 224, 3)
```

Entrée [11]:

```
gold_fish[np.newaxis, ...]
```

Out[11]:

```
array([[[[0.28235294, 0.33333333, 0.07058824],
         [0.31372549, 0.37254902, 0.09019608],
         [0.34901961, 0.41960784, 0.11764706],
         ...,
         [0.32941176, 0.39215686, 0.00392157],
         [0.32156863, 0.38431373, 0.00392157],
         [0.30980392, 0.36862745, 0.        ]],

        [[0.28627451, 0.33333333, 0.08235294],
         [0.3254902 , 0.38039216, 0.10980392],
         [0.35294118, 0.42352941, 0.12941176],
         ...,
         [0.32156863, 0.38039216, 0.00392157],
         [0.31372549, 0.37254902, 0.00392157],
         [0.30196078, 0.36078431, 0.        ]],

        [[0.28627451, 0.33333333, 0.08627451],
         [0.31372549, 0.36862745, 0.10196078],
         [0.34509804, 0.41568627, 0.12941176],
         ...,
         [0.31764706, 0.37647059, 0.00392157],
         [0.30980392, 0.36862745, 0.00784314],
         [0.29803922, 0.35686275, 0.00392157]],

        ...,

        [[0.05490196, 0.10980392, 0.01568627],
         [0.05098039, 0.11372549, 0.01960784],
         [0.05098039, 0.12156863, 0.02352941],
         ...,
         [0.15686275, 0.21960784, 0.03921569],
         [0.15686275, 0.22352941, 0.03529412],
         [0.16078431, 0.22352941, 0.03137255]],

        [[0.0627451 , 0.1254902 , 0.01568627],
         [0.05882353, 0.13333333, 0.01960784],
         [0.05490196, 0.1372549 , 0.01960784],
         ...,
         [0.1372549 , 0.20392157, 0.04705882],
         [0.14117647, 0.20784314, 0.04313725],
         [0.14117647, 0.20784314, 0.03529412]],

        [[0.06666667, 0.14509804, 0.01176471],
         [0.07058824, 0.15294118, 0.01960784],
         [0.05490196, 0.14901961, 0.01176471],
         ...,
         [0.11372549, 0.18039216, 0.04313725],
         [0.11764706, 0.18431373, 0.03921569],
         [0.11764706, 0.18823529, 0.03529412]]]])
```

Entrée [12]:

```
result = classifier.predict(gold_fish[np.newaxis, ...])
result.shape
```

```
1/1 [==============================] - 6s 6s/step
```

Out[12]:

```
(1, 1001)
```

Entrée [13]:

```
predicted_label_index = np.argmax(result)
predicted_label_index
```

Out[13]:

```
2
```

Entrée [15]:

```
tf.keras.utils.get_file('ImageNetLabels.txt','https://storage.googleapis.com/download.tenso
```

Out[15]:

```
'C:\\Users\\Iliess Zahid\\.keras\\datasets\\ImageNetLabels.txt'
```

Load flowers dataset

Entrée [19]:

```
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower
data_dir = tf.keras.utils.get_file('flower_photos', origin=dataset_url,  cache_dir='.', unt
# cache_dir indicates where to download data. I specified . which means current directory
# untar true will unzip it
```

Entrée [20]:

```
data_dir
```

Out[20]:

```
'.\\datasets\\flower_photos'
```

Entrée [21]:

```
import pathlib
data_dir = pathlib.Path(data_dir)
data_dir
```

Out[21]:

```
WindowsPath('datasets/flower_photos')
```

Entrée [22]:

```
list(data_dir.glob('*/*.jpg'))[:5]
```

Out[22]:

```
[WindowsPath('datasets/flower_photos/daisy/100080576_f52e8ee070_n.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10140303196_b88d3d6cec.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172379554_b296050f82_n.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172567486_2748826a8b.jpg'),
 WindowsPath('datasets/flower_photos/daisy/10172636503_21bededa75_n.jpg')]
```

Entrée [23]:

```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
3670
```

Entrée [24]:

```
roses = list(data_dir.glob('roses/*'))
roses[:5]
```

Out[24]:

```
[WindowsPath('datasets/flower_photos/roses/10090824183_d02c613f10_m.jpg'),
 WindowsPath('datasets/flower_photos/roses/102501987_3cdb8e5394_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/10503217854_e66a804309.jpg'),
 WindowsPath('datasets/flower_photos/roses/10894627425_ec76bbc757_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/110472418_87b6a3aa98_m.jpg')]
```

Entrée [26]:

```
import PIL
PIL.Image.open(str(roses[1]))
```

Out[26]:

Entrée [27]:

```
tulips = list(data_dir.glob('tulips/*'))
PIL.Image.open(str(tulips[0]))
```

Out[27]:

Read flowers images from disk into numpy array using opencv

Entrée [28]:

```
flowers_images_dict = {
    'roses': list(data_dir.glob('roses/*')),
    'daisy': list(data_dir.glob('daisy/*')),
    'dandelion': list(data_dir.glob('dandelion/*')),
    'sunflowers': list(data_dir.glob('sunflowers/*')),
    'tulips': list(data_dir.glob('tulips/*')),
}
```

Entrée [29]:

```
flowers_labels_dict = {
    'roses': 0,
    'daisy': 1,
    'dandelion': 2,
    'sunflowers': 3,
    'tulips': 4,
}
```

Entrée [30]:

```
flowers_images_dict['roses'][:5]
```

Out[30]:

```
[WindowsPath('datasets/flower_photos/roses/10090824183_d02c613f10_m.jpg'),
 WindowsPath('datasets/flower_photos/roses/102501987_3cdb8e5394_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/10503217854_e66a804309.jpg'),
 WindowsPath('datasets/flower_photos/roses/10894627425_ec76bbc757_n.jpg'),
 WindowsPath('datasets/flower_photos/roses/110472418_87b6a3aa98_m.jpg')]
```

Entrée [31]:

```
str(flowers_images_dict['roses'][0])
```

Out[31]:

```
'datasets\\flower_photos\\roses\\10090824183_d02c613f10_m.jpg'
```

Entrée [32]:

```
img = cv2.imread(str(flowers_images_dict['roses'][0]))
```

Entrée [33]:

```
img.shape
```

Out[33]:

```
(240, 179, 3)
```

Entrée [34]:

```
cv2.resize(img,(224,224)).shape
```

Out[34]:

```
(224, 224, 3)
```

Entrée [35]:

```
X, y = [], []

for flower_name, images in flowers_images_dict.items():
    for image in images:
        img = cv2.imread(str(image))
        resized_img = cv2.resize(img,(224,224))
        X.append(resized_img)
        y.append(flowers_labels_dict[flower_name])
```

Entrée [36]:

```
X = np.array(X)
y = np.array(y)
```

Train test split

Entrée [38]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Preprocessing: scale images

Entrée [39]:

```
X_train_scaled = X_train / 255
X_test_scaled = X_test / 255
```

Make prediction using pre-trained model on new flowers dataset

Entrée [40]:

```
X[0].shape
```

Out[40]:

```
(224, 224, 3)
```

Entrée [41]:

```
IMAGE_SHAPE+(3,)
```

Out[41]:

```
(224, 224, 3)
```

Entrée [42]:

```
x0_resized = cv2.resize(X[0], IMAGE_SHAPE)
x1_resized = cv2.resize(X[1], IMAGE_SHAPE)
x2_resized = cv2.resize(X[2], IMAGE_SHAPE)
```

Entrée [43]:

```
plt.axis('off')
plt.imshow(X[0])
```

Out[43]:

```
<matplotlib.image.AxesImage at 0x1a3a6211e50>
```

Entrée [44]:

```
predicted = classifier.predict(np.array([x0_resized, x1_resized, x2_resized]))
predicted = np.argmax(predicted, axis=1)
predicted
```

```
1/1 [==============================] - 27s 27s/step
```

Out[44]:

```
array([795, 880, 795], dtype=int64)
```

Now take pre-trained model and retrain it using flowers images

Entrée [46]:

```
feature_extractor_model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector

pretrained_model_without_top_layer = hub.KerasLayer(
    feature_extractor_model, input_shape=(224, 224, 3), trainable=False)
```

Entrée [47]:

```
num_of_flowers = 5

model = tf.keras.Sequential([
  pretrained_model_without_top_layer,
  tf.keras.layers.Dense(num_of_flowers)
])

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_2 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 5)                 6405

=================================================================
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_________________________________________________________________
```

Entrée [48]:

```
model.compile(
  optimizer="adam",
  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['acc'])

model.fit(X_train_scaled, y_train, epochs=5)
```

```
Epoch 1/5
86/86 [==============================] - 302s 2s/step - loss: 0.8234 - acc:
0.6919
Epoch 2/5
86/86 [==============================] - 112s 1s/step - loss: 0.4218 - acc:
0.8525
Epoch 3/5
86/86 [==============================] - 110s 1s/step - loss: 0.3332 - acc:
0.8819
Epoch 4/5
86/86 [==============================] - 110s 1s/step - loss: 0.2776 - acc:
0.9128
Epoch 5/5
86/86 [==============================] - 110s 1s/step - loss: 0.2353 - acc:
0.9324
```

Out[48]:

```
<keras.callbacks.History at 0x1a517b005b0>
```

Entrée [ ]: