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

```
# Install tensorflow_hub using pip install tensorflow_hub first
```

```
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)

```
IMAGE_SHAPE = (224, 224)

classifier = tf.keras.Sequential([
    hub.KerasLayer("https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4", input_shape=IMAGE_SHAPE+(3,))
])
```

```
gold_fish = Image.open("gold-fish-1.jpg").resize(IMAGE_SHAPE)
gold_fish
```

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

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

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

```
array([[[[0.01176471, 0.03529412, 0.        ],
         [0.01960784, 0.04313725, 0.        ],
         [0.02352941, 0.04705882, 0.        ],
         ...,
         [0.07058824, 0.08235294, 0.        ],
         [0.07058824, 0.08235294, 0.        ],
         [0.07058824, 0.08235294, 0.        ]],

        [[0.01176471, 0.03529412, 0.        ],
         [0.01960784, 0.04313725, 0.        ],
         [0.02745098, 0.04705882, 0.        ],
         ...,
         [0.07058824, 0.08235294, 0.        ],
         [0.07058824, 0.08235294, 0.        ],
         [0.07058824, 0.08235294, 0.        ]],

        [[0.01176471, 0.03529412, 0.        ],
         [0.01960784, 0.04313725, 0.        ],
         [0.02745098, 0.05098039, 0.00392157],
         ...,
         [0.07058824, 0.08235294, 0.        ],
         [0.07058824, 0.08235294, 0.        ],
         [0.07058824, 0.08235294, 0.        ]],
```

```
        ...,

        [[0.01176471, 0.01176471, 0.00392157],
         [0.01176471, 0.01176471, 0.00392157],
         [0.01176471, 0.01176471, 0.00392157],
         ...,
         [0.04705882, 0.04705882, 0.00784314],
         [0.04313725, 0.04313725, 0.00392157],
         [0.03529412, 0.04313725, 0.        ]],

        [[0.01176471, 0.01176471, 0.00392157],
         [0.01176471, 0.01176471, 0.00392157],
         [0.01176471, 0.01176471, 0.00392157],
         ...,
         [0.04705882, 0.04705882, 0.00392157],
         [0.04313725, 0.04313725, 0.00392157],
         [0.03529412, 0.04313725, 0.        ]],

        [[0.00784314, 0.00784314, 0.        ],
         [0.00784314, 0.00784314, 0.        ],
         [0.01176471, 0.01176471, 0.00392157],
         ...,
         [0.04705882, 0.04705882, 0.        ],
         [0.04313725, 0.04313725, 0.        ],
         [0.03529412, 0.04313725, 0.        ]]])
```

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

```
1/1 [==============================] - 1s 872ms/step
(1, 1001)
```

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

```
2
```

```
#tf.keras.utils.get_file('ImageNetLabels.txt','https://storage.googleapis.com/download.tensorflow.org/data/ImageNetLabels.txt')
image_labels = []
with open("ImageNetLabels.txt", "r") as f:
    image_labels = f.read().splitlines()
image_labels[:5]
```

```
['background', 'tench', 'goldfish', 'great white shark', 'tiger shark']
```

```
image_labels[predicted_label_index]
```

```
'goldfish'
```

Load flowers dataset

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

```
data_dir
```

```
'./datasets/flower_photos'
```

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

```
PosixPath('datasets/flower_photos')
```

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

```
[PosixPath('datasets/flower_photos/daisy/6323721068_3d3394af6d_n.jpg'),
 PosixPath('datasets/flower_photos/daisy/144076848_57e1d662e3_m.jpg'),
 PosixPath('datasets/flower_photos/daisy/16020253176_60f2a6a5ca_n.jpg'),
 PosixPath('datasets/flower_photos/daisy/422094774_28acc69a8b_n.jpg'),
 PosixPath('datasets/flower_photos/daisy/8708143485_38d084ac8c_n.jpg')]
```

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

```
3670
```

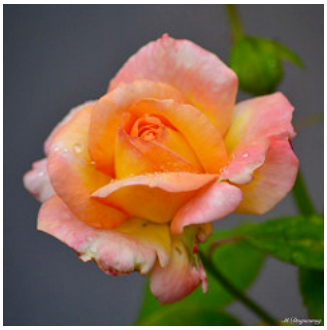
```
roses = list(data_dir.glob('roses/*'))
roses[:5]
```

```
[PosixPath('datasets/flower_photos/roses/14019883858_e5d2a0ec10_n.jpg'),
 PosixPath('datasets/flower_photos/roses/20825078671_90b0389c70_m.jpg'),
 PosixPath('datasets/flower_photos/roses/17158274118_00ec99a23c.jpg'),
 PosixPath('datasets/flower_photos/roses/4713533500_fcc295de70_n.jpg'),
 PosixPath('datasets/flower_photos/roses/5419629292_2f06e4b295.jpg')]
```

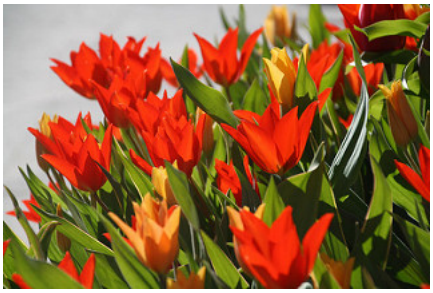
```
pip install pillow
```

```
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
```

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



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



Read flowers images from disk into numpy array using opencv

```
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/*')),
}
```

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

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

```
[PosixPath('datasets/flower_photos/roses/14019883858_e5d2a0ec10_n.jpg'),
 PosixPath('datasets/flower_photos/roses/20825078671_90b0389c70_m.jpg'),
 PosixPath('datasets/flower_photos/roses/17158274118_00ec99a23c.jpg'),
 PosixPath('datasets/flower_photos/roses/4713533500_fcc295de70_n.jpg'),
 PosixPath('datasets/flower_photos/roses/5419629292_2f06e4b295.jpg')]
```

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

```
'datasets/flower_photos/roses/14019883858_e5d2a0ec10_n.jpg'
```

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

```
img.shape
```

```
(231, 320, 3)
```

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

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

```
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])
```

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

Train test split

```
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

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

Make prediction using pre-trained model on new flowers dataset

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

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

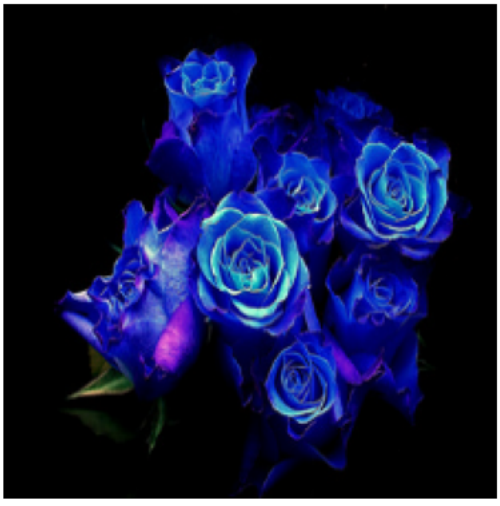
```
IMAGE_SHAPE+(3,)
```

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

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

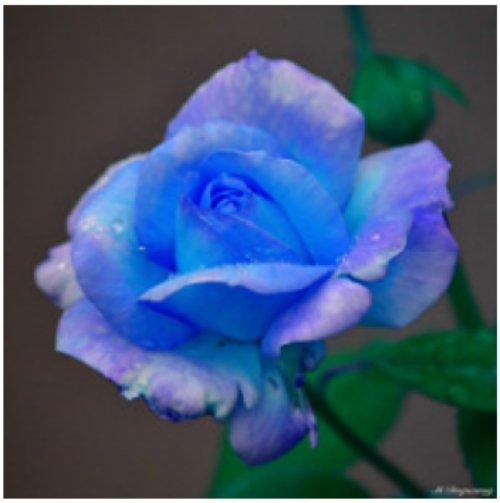
```
plt.axis('off')
plt.imshow(X[0])
```

```
<matplotlib.image.AxesImage at 0x7e632a05ec20>
```



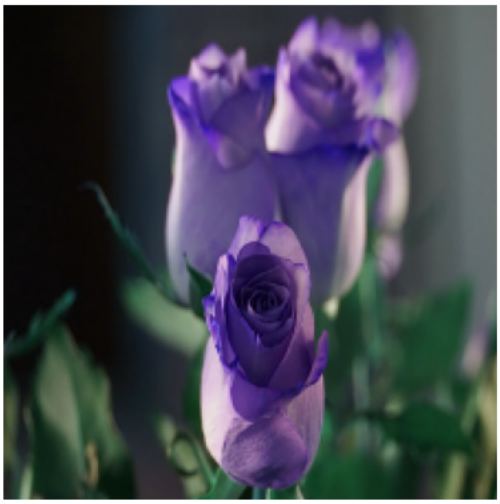
```
plt.axis('off')
plt.imshow(X[1])
```

```
<matplotlib.image.AxesImage at 0x7e632a108340>
```



```
plt.axis('off')
plt.imshow(X[2])
```

```
<matplotlib.image.AxesImage at 0x7e6207bda650>
```



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

```
1/1 [==============================] - 1s 1s/step
array([612, 795, 795])
```

```
image_labels[795]
```

```
'shower curtain'
```

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

```
feature_extractor_model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"

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

```
num_of_flowers = 5

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

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 1280)              2257984

 dense (Dense)               (None, 5)                 6405

=================================================================
Total params: 2264389 (8.64 MB)
Trainable params: 6405 (25.02 KB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

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

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


model.evaluate(X_test_scaled,y_test)
```