USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

Ex No: 5 Date: 4/09/2024	**Transfer learning in image classification**

Objective:

The objective is to demonstrate the application of transfer learning for image classification using TensorFlow. The goal is to leverage pre-trained models from TensorFlow Hub and fine-tune them on a new dataset—in this case, a flower dataset. This approach significantly reduces the time and computational resources required to train a model from scratch while maintaining high accuracy.

Descriptions:

Transfer learning is a machine learning technique where a pre-trained model is adapted to a new but similar problem. This process involves taking a model that has been trained on a large dataset and repurposing it for a different but related task. In this project, we use TensorFlow Hub, a repository of pre-trained models, to perform image classification on a flowers dataset.

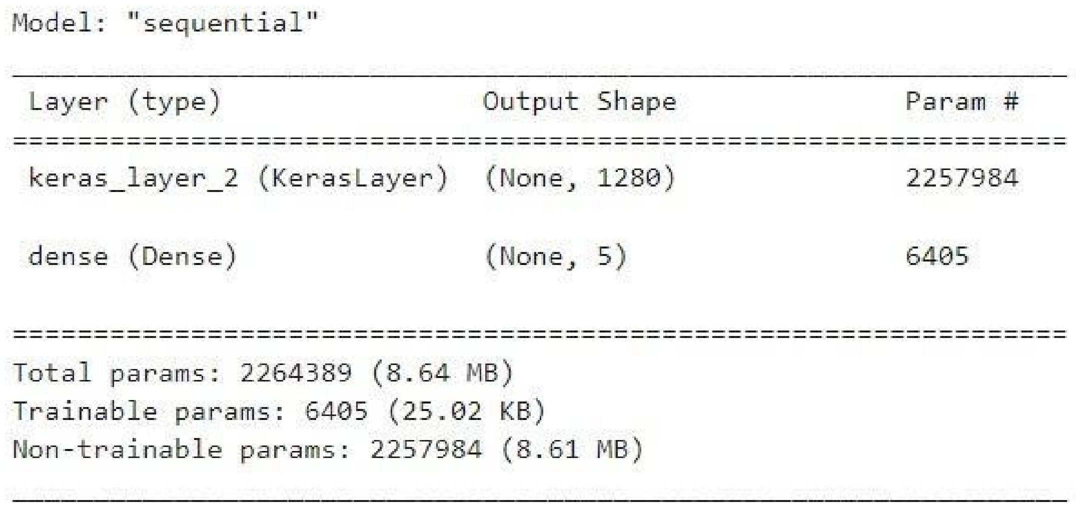
TensorFlow Hub simplifies the process of applying transfer learning by providing a variety of pre-trained models that can be easily integrated into TensorFlow applications. These models, trained on large-scale datasets, can serve as a starting point for new tasks, allowing us to focus on fine-tuning the model rather than training from scratch.

The key steps in this project include:

1. Loading the Pre-trained Model: We begin by selecting a pre-trained image classification model from TensorFlow Hub. This model has already been trained on a large dataset (e.g., ImageNet) and is capable of recognizing a wide variety of images.
2. Preparing the Dataset: The flowers dataset is loaded and preprocessed. This dataset consists of images categorized into different flower types, which will be used to fine-tune the pre-trained model.
3. Model Fine-tuning: The pre-trained model is modified by replacing its final layer with a new layer that matches the number of flower categories. The model is then trained (fine-tuned) on the flowers dataset to adapt it to the new classification task.
4. Evaluation and Testing: After fine-tuning, the model is evaluated on a separate test set to assess its performance. Metrics such as accuracy are used to measure how well the model classifies the flowers.
5. Deployment: Finally, the fine-tuned model can be saved and deployed for practical use, such as in a web application or mobile app for flower recognition.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

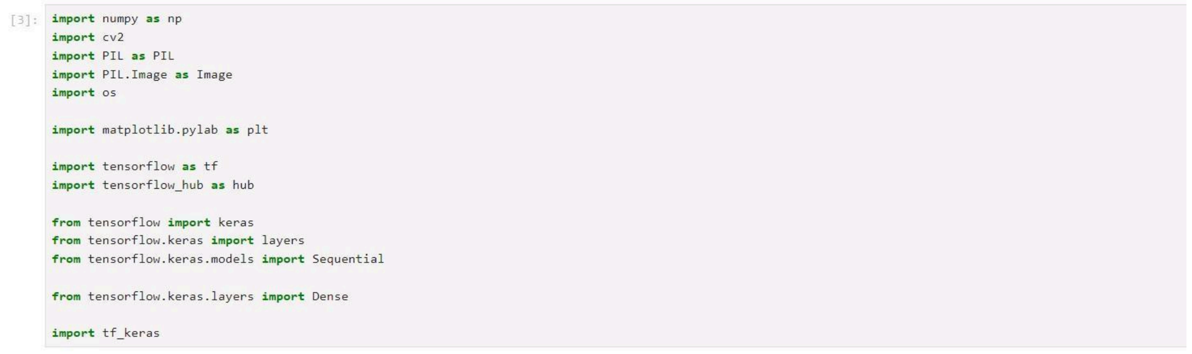
Model:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_2 (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)
_________________________________________________________________
```

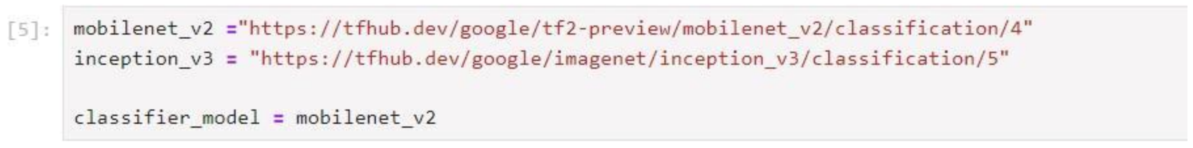
Building the parts of algorithm

Importing Libraries



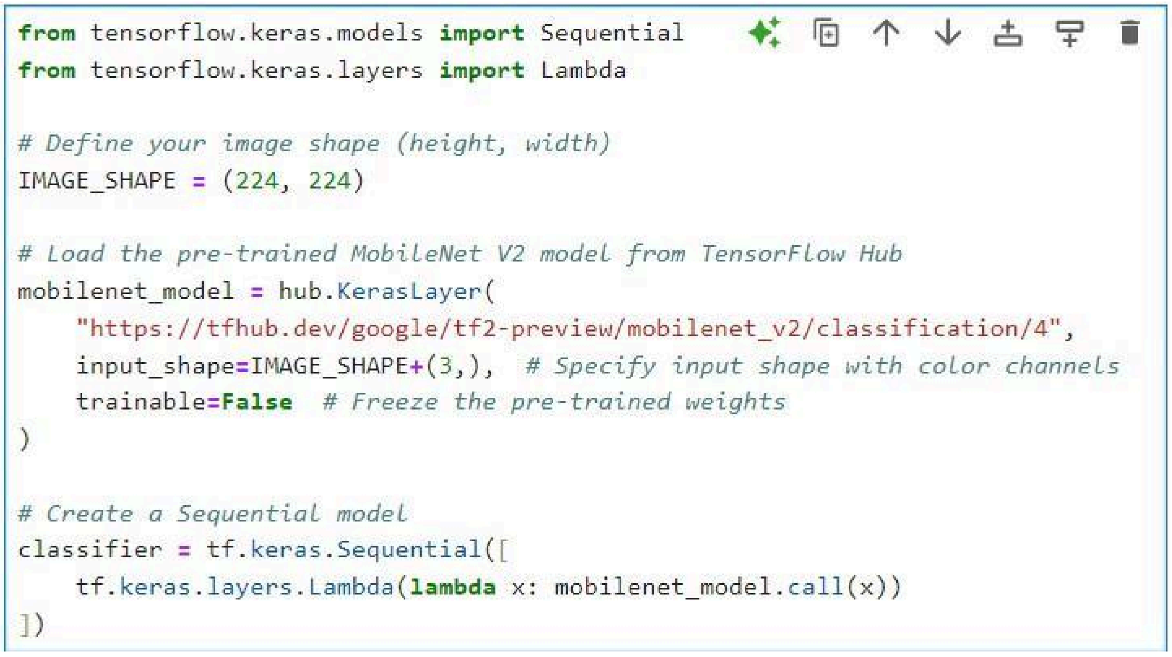
```
[3]: import numpy as np
     import cv2
     import PIL as PIL
     import PIL.Image as Image
     import os

     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

     from tensorflow.keras.layers import Dense

     import tf_keras
```

Download the classifier



```
[5]: mobilenet_v2 ="https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4"
     inception_v3 = "https://tfhub.dev/google/imagenet/inception_v3/classification/5"

     classifier_model = mobilenet_v2
```

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj



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

# Define your image shape (height, width)
IMAGE_SHAPE = (224, 224)

# Load the pre-trained MobileNet V2 model from TensorFlow Hub
mobilenet_model = hub.KerasLayer(
    "https://tfhub.dev/google/tf2-preview/mobilenet_v2/classification/4",
    input_shape=IMAGE_SHAPE+(3,),  # Specify input shape with color channels
    trainable=False  # Freeze the pre-trained weights
)

# Create a Sequential model
classifier = tf.keras.Sequential([
    tf.keras.layers.Lambda(lambda x: mobilenet_model.call(x))
])
```

1. Importing Required Libraries

- TensorFlow Keras Models: The Sequential model allows you to create a linear stack of layers, where you can easily add layers one after another.
- Lambda Layer: This is used to wrap any arbitrary function or computation as a layer, which is useful for adding custom operations within the model.

2. Defining Image Shape

- You specify the size of the images that the model will process. In this case, the images are 224x224 pixels, which is a common size for models like MobileNet.

3. Loading a Pre-trained MobileNet V2 Model

- You load the MobileNet V2 model from TensorFlow Hub, which is a repository of pre-trained models. The model is designed to classify images and is pre-trained on a large dataset.
- The model's weights are frozen (trainable=False), meaning they won't be updated during training, allowing you to use the model as a feature extractor.

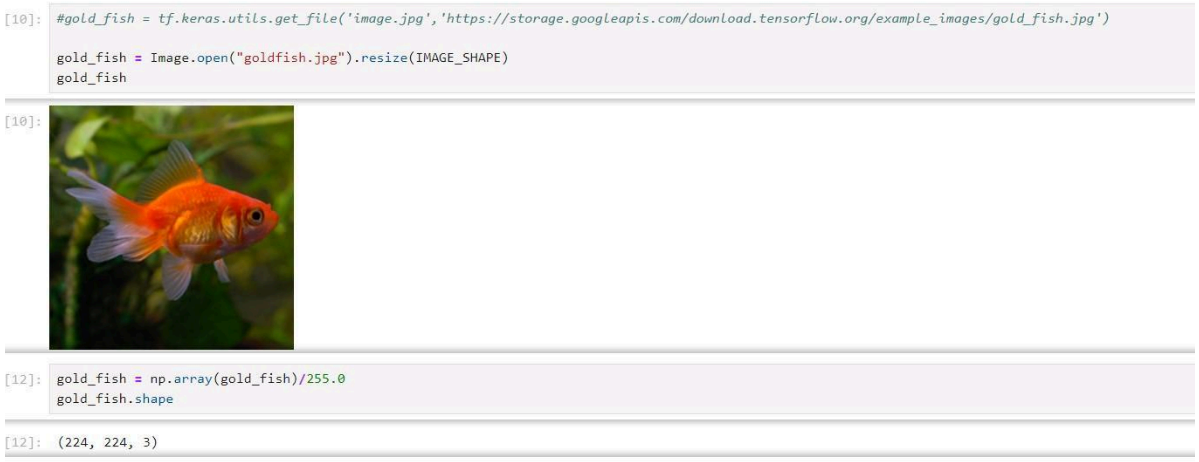
4. Creating a Sequential Model

- A new model is created using TensorFlow's Sequential API.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

- A Lambda layer is used to call the pre-trained MobileNet model. This layer takes the input images, processes them through the MobileNet model, and outputs the features or classifications produced by MobileNet.

Run it on one image



```
[10]: #gold_fish = tf.keras.utils.get_file('image.jpg','https://storage.googleapis.com/download.tensorflow.org/example_images/gold_fish.jpg')

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

```
[10]:
```

```
[12]: gold_fish = np.array(gold_fish)/255.0
      gold_fish.shape
```

```
[12]: (224, 224, 3)
```

Open and Resize Image:

- Load the image file goldfish.jpg.
- Resize the image to the dimensions specified by IMAGE_SHAPE.

Display Image Object:

- Display the image object (useful for visual inspection in interactive environments).

Convert to NumPy Array and Normalize:

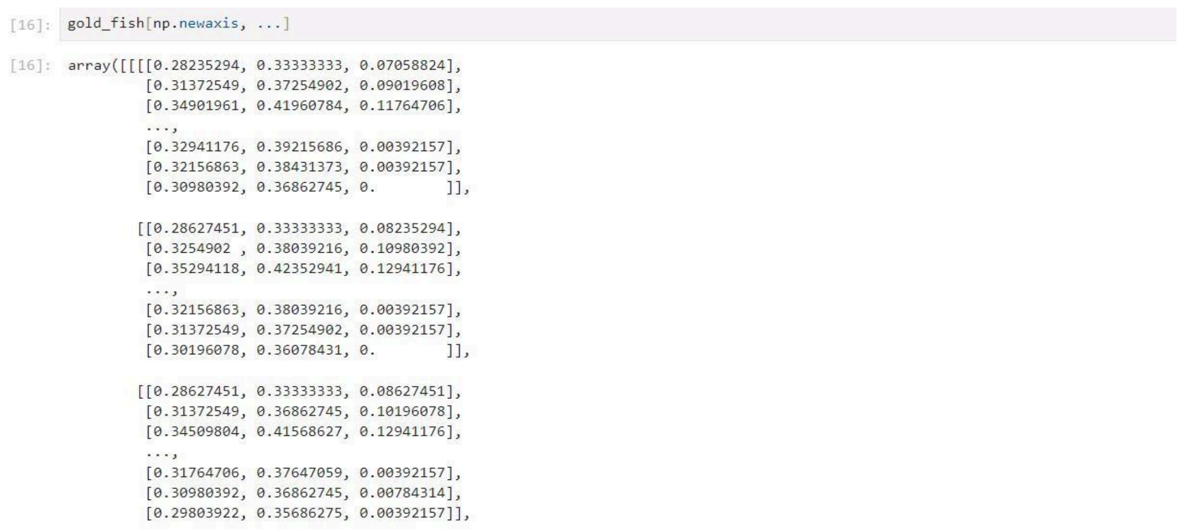
- Convert the resized image into a NumPy array.
- Normalize the pixel values from the range [0, 255] to [0, 1].

Check Array Shape:

- Print or check the shape of the NumPy array representing the image.
- The shape provides the dimensions of the image and the number of color channels.

Add a batch dimension (with np.newaxis)

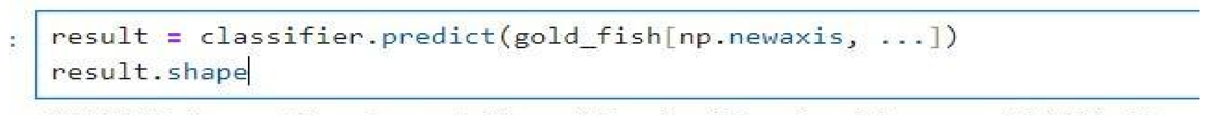
USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj



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

[16]: 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]],
```

- Current Array Shape: Suppose gold_fish has a shape of (height, width, channels), where height and width are the dimensions of the image, and channels represents the color channels (e.g., RGB).
- New Shape with np.newaxis: By using gold_fish[np.newaxis, ...], you add a new axis at the beginning of the array. This changes the shape to (1, height, width, channels).
- Purpose: This new shape is often used for batching, where the first dimension represents the batch size. For instance, in machine learning frameworks, models usually expect inputs with a batch dimension, even if the batch size is 1.

So, applying gold_fish[np.newaxis, ...] transforms a single image array into a format suitable for processing in models that expect batches of images.



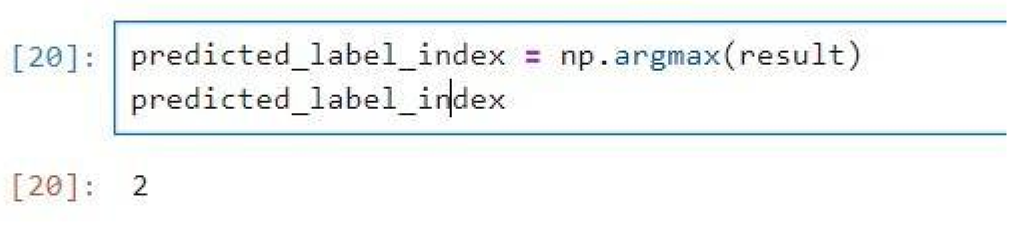
```
: result = classifier.predict(gold_fish[np.newaxis, ...])
  result.shape
```

Input to predict: gold_fish[np.newaxis, ...] adds a batch dimension to your image, changing its shape from (height, width, channels) to (1, height, width, channels). This is required because many classifiers expect input data in batches, even if there is only one image. Prediction Output: The predict method of a classifier will return the predictions for the input data. The shape of result depends on the classifier and the type of problem you are solving:

- For Classification: If you're doing classification, result will typically have the shape (1, num_classes), where num_classes is the number of classes in your classification problem. The 1 represents the single image in the batch, and num_classes represents the predicted probabilities or scores for each class.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

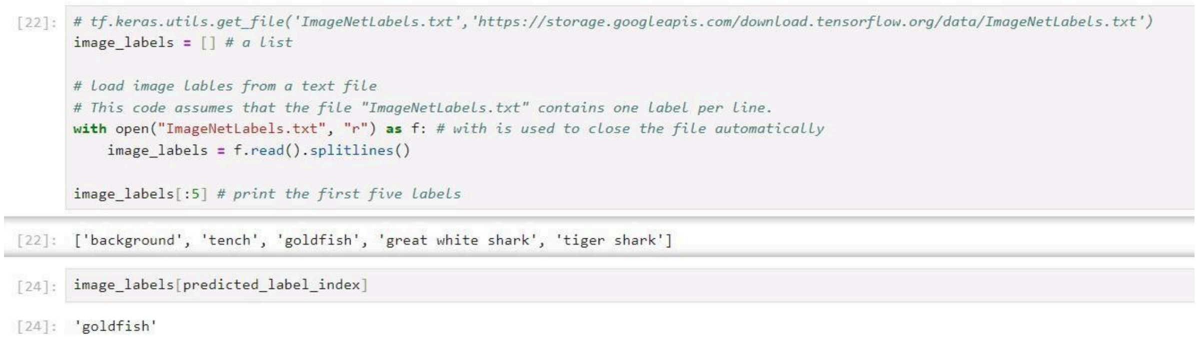
- For Other Types of Models: If you're using a different kind of model (e.g., regression, object detection), the shape of result will vary according to the model's output.



```
[20]: predicted_label_index = np.argmax(result)
      predicted_label_index

[20]: 2
```

predicted_label_index will be an integer representing the index of the highest value in the result array.

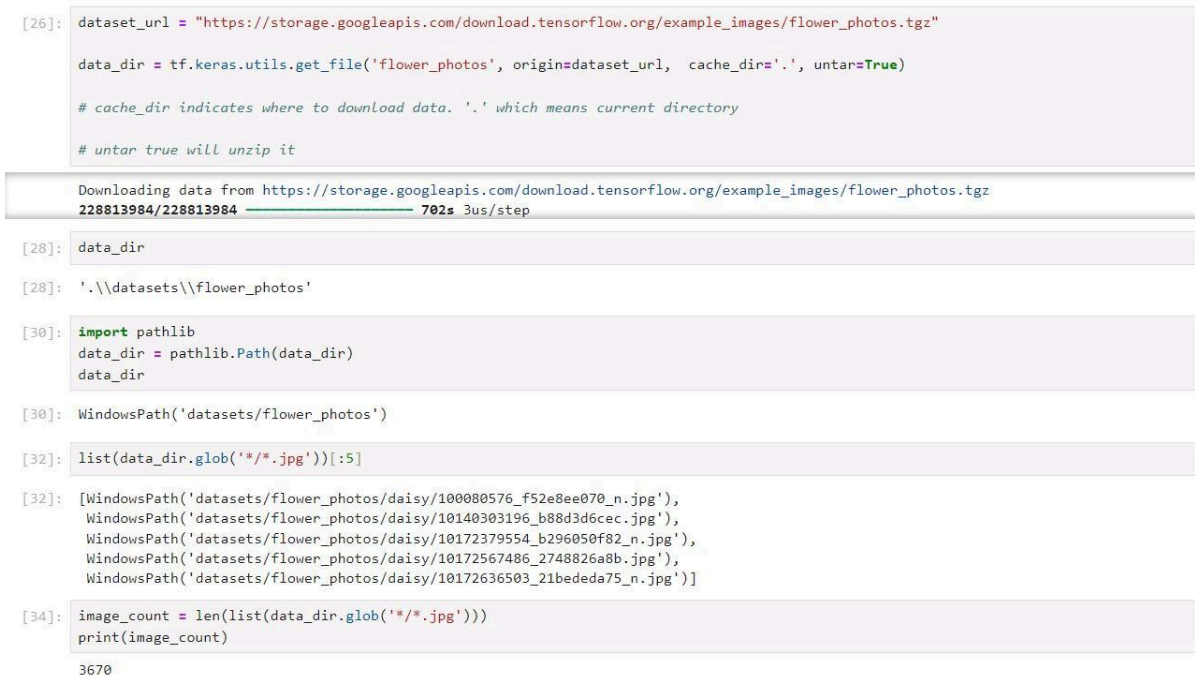
This index corresponds to the predicted class label.



```
[22]: # tf.keras.utils.get_file('ImageNetLabels.txt','https://storage.googleapis.com/download.tensorflow.org/data/ImageNetLabels.txt')
      image_labels = [] # a list

      # load image lables from a text file
      # This code assumes that the file "ImageNetLabels.txt" contains one label per line.
      with open("ImageNetLabels.txt", "r") as f: # with is used to close the file automatically
          image_labels = f.read().splitlines()

      image_labels[:5] # print the first five labels

[22]: ['background', 'tench', 'goldfish', 'great white shark', 'tiger shark']

[24]: image_labels[predicted_label_index]

[24]: 'goldfish'
```

- Uncomment the file download line if needed.
- Initialize an empty list for labels.
- Open and read the ImageNetLabels.txt file.
- Split the file content into a list of labels. ● Print the first five labels.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj



```
[26]: 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. '.' which means current directory

      # untar true will unzip it

      Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
      228813984/228813984 ━━━━━━━━━━━━━━━━━━━━ 702s 3us/step

[28]: data_dir

[28]: '.\\datasets\\flower_photos'

[30]: import pathlib
      data_dir = pathlib.Path(data_dir)
      data_dir

[30]: WindowsPath('datasets/flower_photos')

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

[32]: [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')]

[34]: image_count = len(list(data_dir.glob('*/*.jpg')))
      print(image_count)

      3670
```

- Define the URL for the dataset.
- Use tf.keras.utils.get_file to download and extract the dataset.
- Import the pathlib module.
- Convert the dataset directory path to a Path object.
- List and display the first 5 image paths.
- Count and print the total number of images in the dataset.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

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

```
[36]: [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')]
```

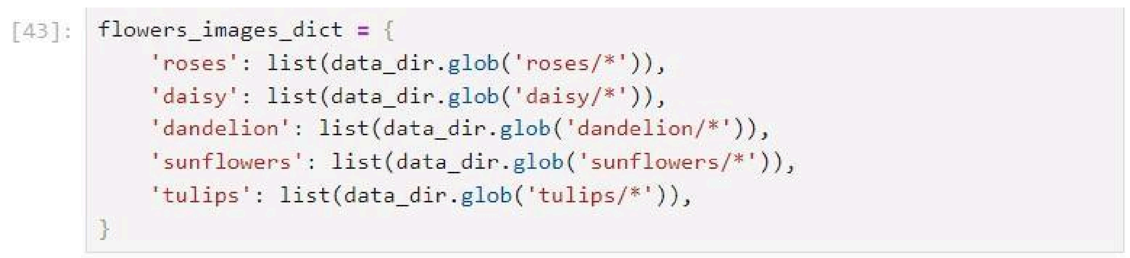
```
[38]: PIL.Image.open(str(roses[1]))
```

[38]:

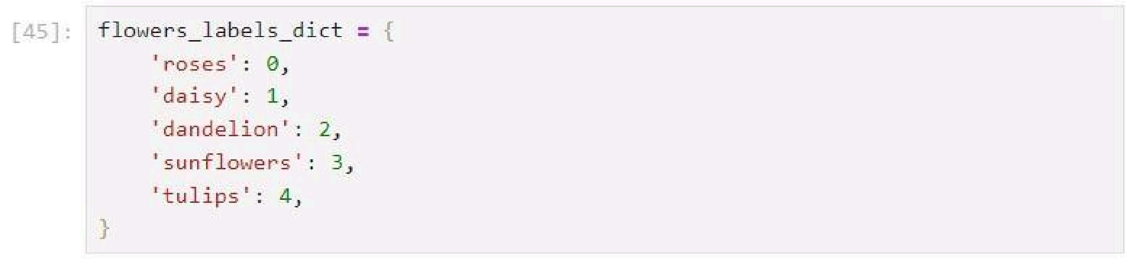
```
[40]: tulips = list(data_dir.glob('tulips/*'))
      PIL.Image.open(str(tulips[0]))
```

- Get a list of all image files in the 'roses' subdirectory.
- Display the first 5 images in the 'roses' list.
- Open and view the second image in the 'roses' list using PIL.

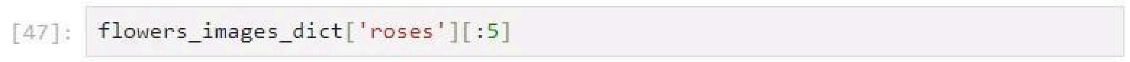
USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj



```
[43]: 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/*')),
      }
```



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

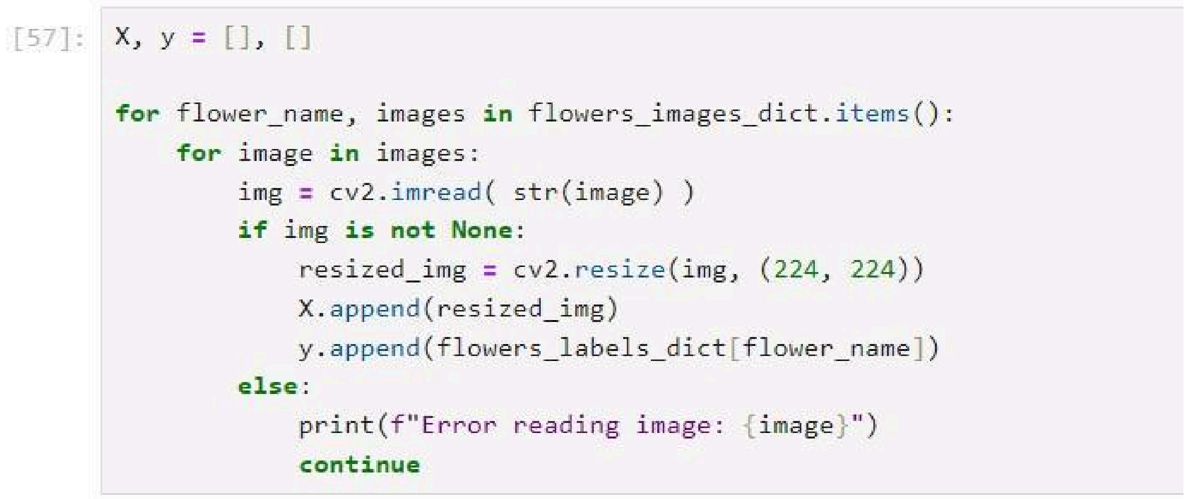


```
[47]: flowers_images_dict['roses'][:5]
```

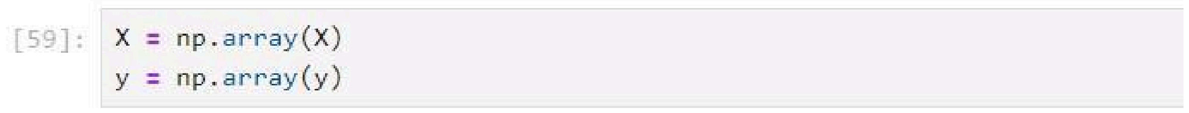
flowers_images_dict:

- A dictionary mapping flower categories to lists of image file paths.
- Keys are flower types (e.g., 'roses', 'daisy').
- Values are lists of image file paths for each flower type. flowers_labels_dict:

- A dictionary mapping flower categories to numerical labels.
- Keys are flower types (e.g., 'roses', 'daisy').
- Values are numerical labels (e.g., 0 for 'roses', 1 for 'daisy').



```
[57]: X, y = [], []

      for flower_name, images in flowers_images_dict.items():
          for image in images:
              img = cv2.imread( str(image) )
              if img is not None:
                  resized_img = cv2.resize(img, (224, 224))
                  X.append(resized_img)
                  y.append(flowers_labels_dict[flower_name])
              else:
                  print(f"Error reading image: {image}")
                  continue
```



```
[59]: X = np.array(X)
      y = np.array(y)
```

Initialize X and y lists.

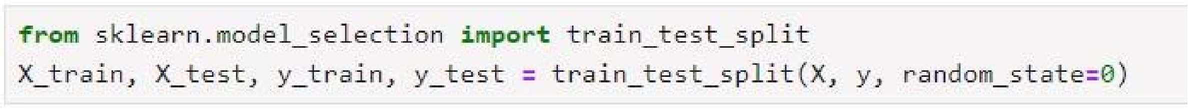
USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

Loop through each flower category:

- Read image with cv2.imread().
- Resize image to (224, 224) with cv2.resize().
- Append resized image to X. ● Append label to y.

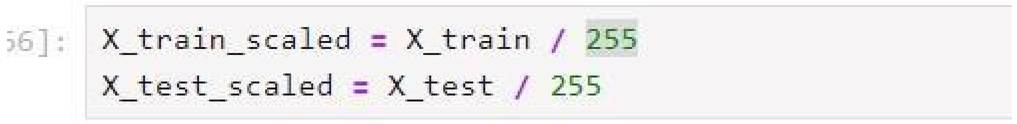
Convert X and y to NumPy arrays.

Handle errors if images can't be read.



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

- Import train_test_split from sklearn.model_selection.
- Split the dataset into training and testing sets using train_test_split.
- Set a random seed (random_state=0) for reproducibility. ● Store the results in X_train, X_test, y_train, and y_test.



```
6]:  X_train_scaled = X_train / 255
     X_test_scaled = X_test / 255
```

- Scale X_train by dividing by 255.
- Store the result in X_train_scaled.
- X_test_scaled is already scaled.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

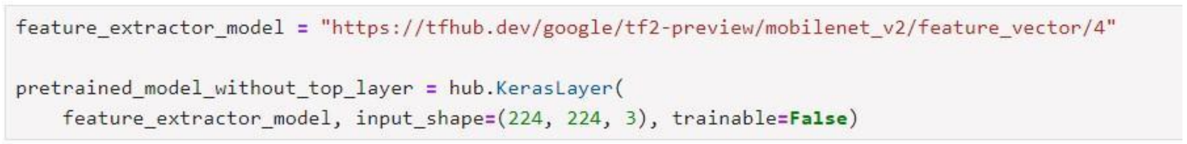
```
[70]: IMAGE_SHAPE+(3,)

[70]: (224, 224, 3)

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

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

[74]: <matplotlib.image.AxesImage at 0x2aedafddd60>
```

- Define IMAGE_SHAPE as (224, 224, 3).
- Resize X[0], X[1], and X[2] to IMAGE_SHAPE using cv2.resize.
- Display X[0] using plt.imshow().
- Use plt.axis('off') to turn off the axis.

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

      1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step

[80]: array([795, 880, 795], dtype=int64)

[82]: image_labels[795]

[82]: 'shower curtain'
```

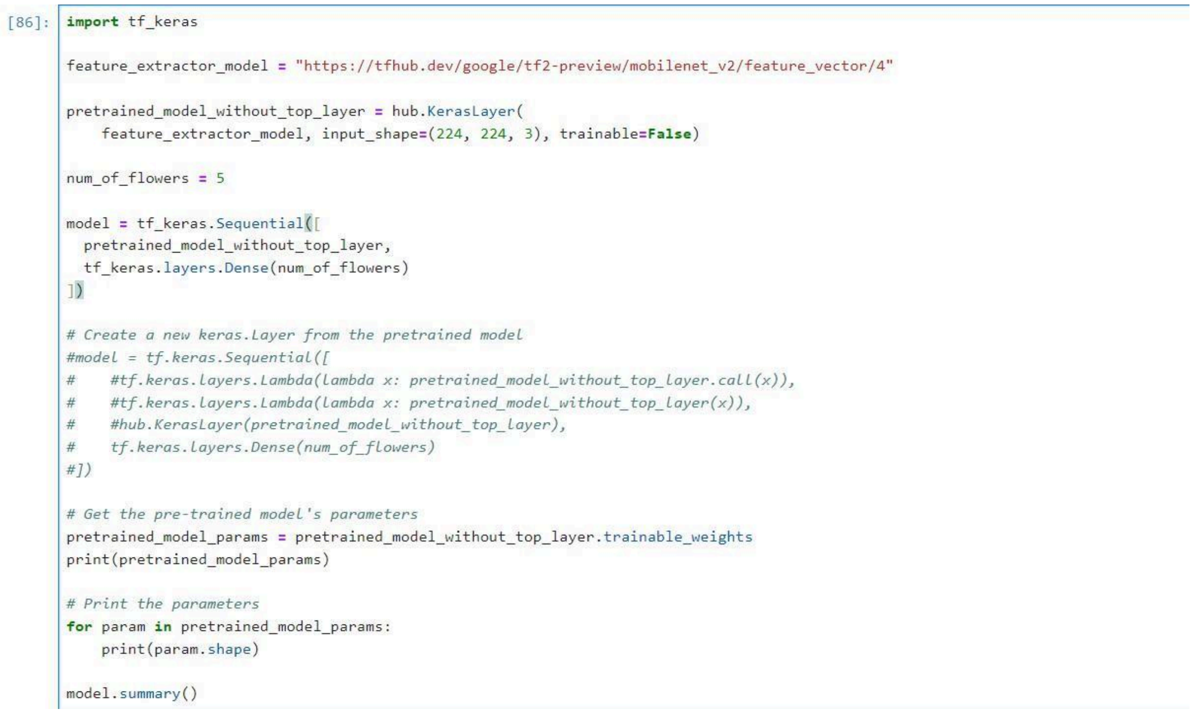
USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

- Predict labels for the resized images x0_resized, x1_resized, and x2_resized using classifier.predict().
- Find the index of the maximum value in the prediction result using np.argmax(). ● Retrieve the label corresponding to the index 795 from image_labels.



```
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)
```

- Define the URL of the feature extractor model: https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4.
- Create a KerasLayer from TensorFlow Hub with the specified model URL.
- Set the input_shape to (224, 224, 3) and set trainable=False to freeze the weights.

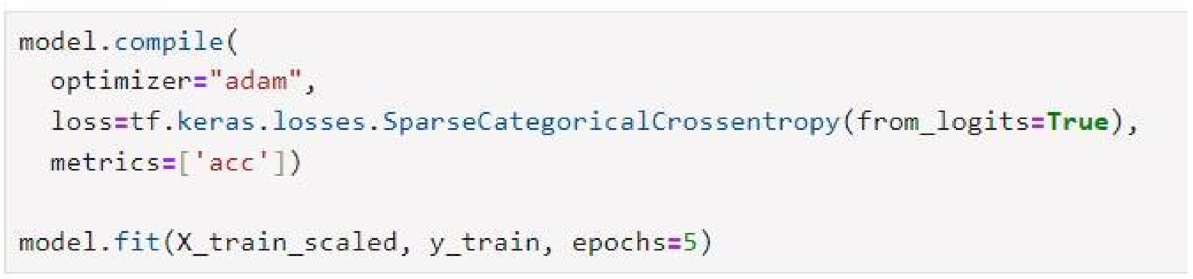


```
[86]: import tf_keras

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)
])

# Create a new keras.Layer from the pretrained model
#model = tf.keras.Sequential([
#    #tf.keras.layers.Lambda(lambda x: pretrained_model_without_top_layer.call(x)),
#    #tf.keras.layers.Lambda(lambda x: pretrained_model_without_top_layer(x)),
#    #hub.KerasLayer(pretrained_model_without_top_layer),
#    tf.keras.layers.Dense(num_of_flowers)
#])

# Get the pre-trained model's parameters
pretrained_model_params = pretrained_model_without_top_layer.trainable_weights
print(pretrained_model_params)

# Print the parameters
for param in pretrained_model_params:
    print(param.shape)

model.summary()
```

- Import TensorFlow and TensorFlow Hub.
- Define the URL of the feature extractor model: "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4".
- Create a KerasLayer from TensorFlow Hub for the MobileNetV2 feature extractor, with input_shape=(224, 224, 3) and trainable=False.
- Define the number of flower categories: num_of_flowers = 5.
- Build a Sequential model with:
    - ○ The feature extractor as the first layer.
    - ○ A dense layer with num_of_flowers units.

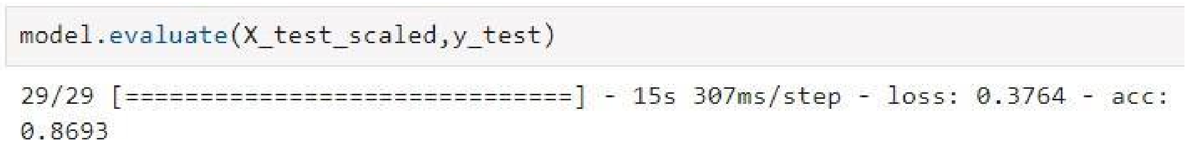
USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj

- Retrieve and print the trainable parameters of the feature extractor.
- Print the shape of each parameter.
- Display the model summary using model.summary().



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

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

- Optimizer: Adam
- Loss: SparseCategoricalCrossentropy (from_logits=True)
- Metrics: Accuracy
- Data: X_train_scaled
- Labels: y_train
- Epochs: 5

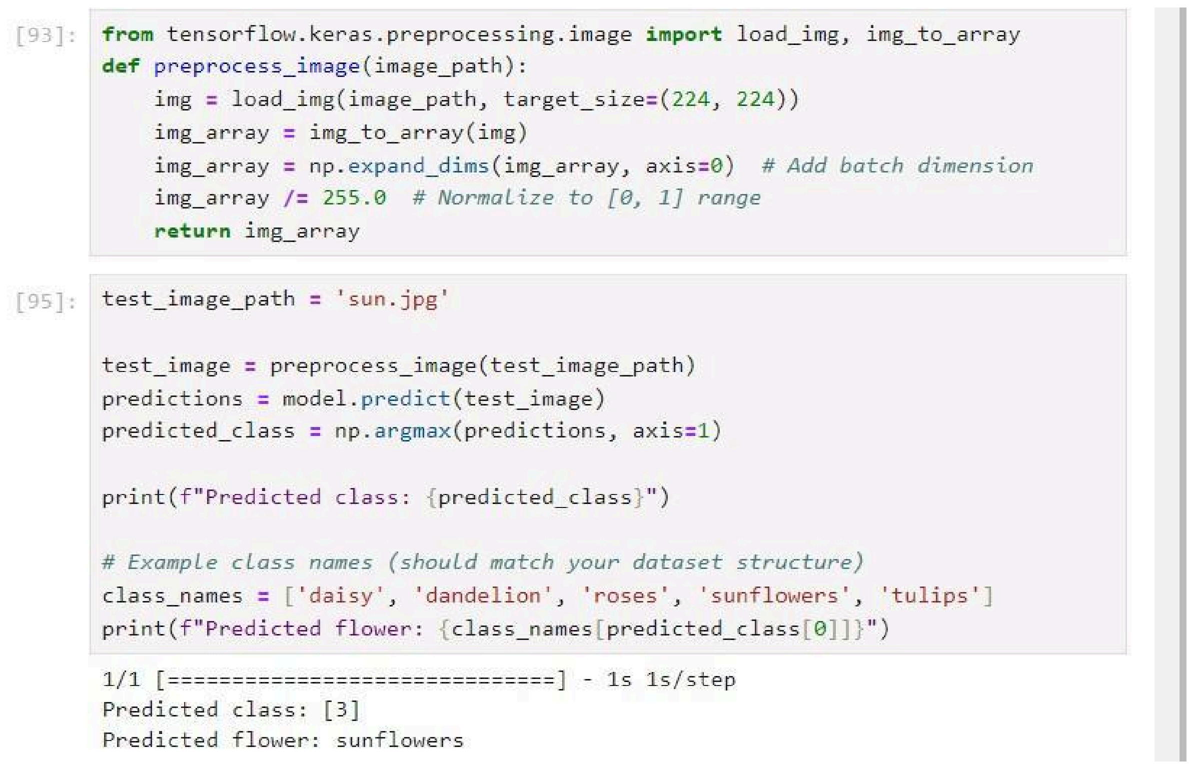


```
model.evaluate(X_test_scaled,y_test)
```

```
29/29 [==============================] - 15s 307ms/step - loss: 0.3764 - acc:
0.8693
```

- Evaluate the model on X_test_scaled and y_test.

USN NUMBER: 1RVU22CSE071
NAME: Joselyn Riana Manoj


```
[93]: from tensorflow.keras.preprocessing.image import load_img, img_to_array
      def preprocess_image(image_path):
          img = load_img(image_path, target_size=(224, 224))
          img_array = img_to_array(img)
          img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
          img_array /= 255.0  # Normalize to [0, 1] range
          return img_array
```

```
[95]: test_image_path = 'sun.jpg'

      test_image = preprocess_image(test_image_path)
      predictions = model.predict(test_image)
      predicted_class = np.argmax(predictions, axis=1)

      print(f"Predicted class: {predicted_class}")

      # Example class names (should match your dataset structure)
      class_names = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
      print(f"Predicted flower: {class_names[predicted_class[0]]}")
```

```
1/1 [==============================] - 1s 1s/step
Predicted class: [3]
Predicted flower: sunflowers
```

- Define preprocess_image() function to:
- Load an image and resize it to (224, 224).
- Convert the image to an array and add a batch dimension.
- Normalize the image array to [0, 1] range.
- Load a test image (test_image_path = 'sun.jpg').
- Preprocess the test image using preprocess_image().
- Predict the class of the test image with the model.
- Determine the predicted class index and print it.
- Define class_names list matching the dataset classes.
- Print the predicted flower class name from class_names.

GitHub Link:
https://github.com/Akshathbtech/Deep-Learning-Labs