```
!pip install -q tensorflow-datasets tensorflow matplotlib
```

```
import numpy as np
import tensorflow as tf
```

```
import tensorflow_datasets as tfds
```

```
from tensorflow.keras.utils import to_categorical
```

```
## Loading images and labels
(train_ds, train_labels), (test_ds, test_labels) = tfds.load("tf_flowers",
    split=["train[:70%]", "train[:30%]"], ## Train test split
    batch_size=-1,
    as_supervised=True,  # Include labels
)
```

```
Downloading and preparing dataset 218.21 MiB (download: 218.21 MiB, generated: 221.83 MiB, total: 440.05 MiB) to /root/tensorflow_datasets/tf_flowers/3.0.1...
Dl Completed...: 100%  5/5 [00:02<00:00, 2.01 file/s]
Dataset tf_flowers downloaded and prepared to /root/tensorflow_datasets/tf_flowers/3.0.1. Subsequent calls will reuse this data.
```

```
## check existing image size
train_ds[0].shape
```

```
TensorShape([442, 1024, 3])
```

```
## Resizing images
train_ds = tf.image.resize(train_ds, (150, 150))
test_ds = tf.image.resize(test_ds, (150, 150))
```

```
train_labels
```

```
<tf.Tensor: shape=(2569,), dtype=int64, numpy=array([2, 3, 3, ..., 0, 2, 0])>
```

```
## Transforming labels to correct format
train_labels = to_categorical(train_labels, num_classes=5)
test_labels = to_categorical(test_labels, num_classes=5)
```

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
```

```
train_ds[0].shape
```

```
TensorShape([150, 150, 3])
```

```
## Loading VGG16 model
base_model = VGG16(weights="imagenet", include_top=False, input_shape=train_ds[0].shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 0s 0us/step
```

```
## will not train base mode
# Freeze Parameters in model's lower convolutional layers
base_model.trainable = False
```

```
## will not train base mode
# Freeze Parameters in model's lower convolutional layers
base_model.trainable = False
```

```
## Preprocessing input
train_ds = preprocess_input(train_ds)
test_ds = preprocess_input(test_ds)
```

```
## model details
base_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 150, 150, 3)]     0

 block1_conv1 (Conv2D)       (None, 150, 150, 64)      1792

 block1_conv2 (Conv2D)       (None, 150, 150, 64)      36928

 block1_pool (MaxPooling2D)  (None, 75, 75, 64)        0

 block2_conv1 (Conv2D)       (None, 75, 75, 128)       73856

 block2_conv2 (Conv2D)       (None, 75, 75, 128)       147584

 block2_pool (MaxPooling2D)  (None, 37, 37, 128)       0

 block3_conv1 (Conv2D)       (None, 37, 37, 256)       295168
```

```
block3_conv2 (Conv2D)       (None, 37, 37, 256)       590080

block3_conv3 (Conv2D)       (None, 37, 37, 256)       590080

block3_pool (MaxPooling2D)  (None, 18, 18, 256)       0

block4_conv1 (Conv2D)       (None, 18, 18, 512)       1180160

block4_conv2 (Conv2D)       (None, 18, 18, 512)       2359808

block4_conv3 (Conv2D)       (None, 18, 18, 512)       2359808

block4_pool (MaxPooling2D)  (None, 9, 9, 512)         0

block5_conv1 (Conv2D)       (None, 9, 9, 512)         2359808

block5_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

block5_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

block5_pool (MaxPooling2D)  (None, 4, 4, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 0 (0.00 Byte)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

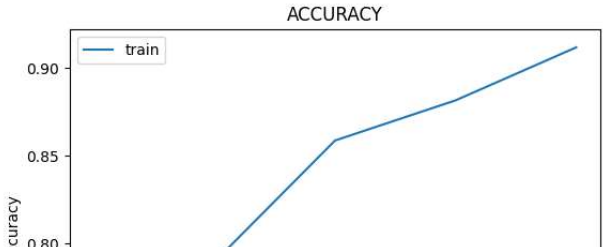
```
#add our layers on top of this model
from tensorflow.keras import layers, models

flatten_layer = layers.Flatten()
dense_layer_1 = layers.Dense(50, activation='relu')
dense_layer_2 = layers.Dense(20, activation='relu')
prediction_layer = layers.Dense(5, activation='softmax')


model = models.Sequential([
    base_model,
    flatten_layer,
    dense_layer_1,
    dense_layer_2,
    prediction_layer
])


from tensorflow.keras.callbacks import EarlyStopping

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy'],
)


es = EarlyStopping(monitor='val_accuracy', mode='max', patience=5,  restore_best_weights=True)


history=model.fit(train_ds, train_labels, epochs=5, validation_split=0.2, batch_size=32, callbacks=[es])
```

```
Epoch 1/5
65/65 [==============================] - 657s 10s/step - loss: 0.8456 - accuracy: 0.7071 - val_loss: 1.2420 - val_accuracy: 0.6304
Epoch 2/5
65/65 [==============================] - 647s 10s/step - loss: 0.5961 - accuracy: 0.7898 - val_loss: 1.0110 - val_accuracy: 0.6790
Epoch 3/5
65/65 [==============================] - 630s 10s/step - loss: 0.3979 - accuracy: 0.8589 - val_loss: 1.1255 - val_accuracy: 0.6984
Epoch 4/5
65/65 [==============================] - 630s 10s/step - loss: 0.3205 - accuracy: 0.8818 - val_loss: 1.1503 - val_accuracy: 0.7101
Epoch 5/5
65/65 [==============================] - 631s 10s/step - loss: 0.2371 - accuracy: 0.9119 - val_loss: 1.3278 - val_accuracy: 0.6946
```

```
los,accurac=model.evaluate(test_ds,test_labels)
print("Loss: ",los,"Accuracy: ", accurac)
```

```
35/35 [==============================] - 291s 8s/step - loss: 0.1887 - accuracy: 0.9292
Loss:  0.18868598341941833 Accuracy:  0.9291552901268005
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.title('ACCURACY')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train'],loc='upper left')
plt.show()
```




```
import numpy as np
import pandas as pd
y_pred = model.predict(test_ds)
y_classes = [np.argmax(element) for element in y_pred]
#to_categorical(y_classes, num_classes=5)
#to_categorical(test_labels, num_classes=5)
print(y_classes[:10])
print("\nTest")
print(test_labels[:10])
```

```
35/35 [==============================] - 262s 7s/step
[2, 3, 3, 2, 3, 0, 0, 0, 0, 4]

Test
[[0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]]
```