```
import tensorflow_datasets as tfds
import tensorflow as tf
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[:60%]", "train[:40%]"], ## 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_da
Dl Completed...: 100%                                                5/5 [00:01<00:00,  2.12 file/s]
```

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

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

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

```
train_labels
```

```
<tf.Tensor: shape=(2202,), dtype=int64, numpy=array([2, 3, 3, ..., 0, 2, 0])>
```

```
train_labels = to_categorical(train_labels, num_classes=5)
test_labels = to_categorical(test_labels, num_classes=5)
```

```
train_labels[0]
```

```
array([0., 0., 1., 0., 0.], dtype=float32)
```

pre trained model using vgg16

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

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

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

```
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_no
58889256/58889256 [==============================] - 0s 0us/step
```

```
base_model.trainable = False
```

```
train_ds = preprocess_input(train_ds)
test_ds = preprocess_input(test_ds)
```

```
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)
_________________________________________________________________
```

```
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)


model.fit(train_ds, train_labels, epochs=50, validation_split=0.2, batch_size=32, callbacks=[es])
```

```
Epoch 1/50
56/56 [==============================] - 589s 10s/step - loss: 1.8304 - accuracy: 0.4600 - val_loss: 1.1535 - val_accuracy: 0.6054
Epoch 2/50
56/56 [==============================] - 586s 11s/step - loss: 0.8380 - accuracy: 0.6928 - val_loss: 0.8972 - val_accuracy: 0.6916
Epoch 3/50
56/56 [==============================] - 607s 11s/step - loss: 0.5583 - accuracy: 0.8064 - val_loss: 0.8641 - val_accuracy: 0.7075
Epoch 4/50
56/56 [==============================] - 553s 10s/step - loss: 0.3588 - accuracy: 0.8790 - val_loss: 1.0272 - val_accuracy: 0.7098
Epoch 5/50
56/56 [==============================] - 582s 10s/step - loss: 0.2540 - accuracy: 0.9154 - val_loss: 0.9202 - val_accuracy: 0.7052
Epoch 6/50
56/56 [==============================] - 580s 10s/step - loss: 0.1714 - accuracy: 0.9500 - val_loss: 0.9346 - val_accuracy: 0.7120
Epoch 7/50
56/56 [==============================] - 551s 10s/step - loss: 0.1168 - accuracy: 0.9659 - val_loss: 1.0789 - val_accuracy: 0.7234
Epoch 8/50
56/56 [==============================] - 581s 10s/step - loss: 0.0981 - accuracy: 0.9682 - val_loss: 1.1942 - val_accuracy: 0.7234
Epoch 9/50
56/56 [==============================] - 581s 10s/step - loss: 0.2117 - accuracy: 0.9245 - val_loss: 1.2767 - val_accuracy: 0.6916
Epoch 10/50
```

```
56/56 [==============================] - 580s 10s/step - loss: 0.2127 - accuracy: 0.9273 - val_loss: 1.3683 - val_accuracy: 0.7052
Epoch 11/50
56/56 [==============================] - 580s 10s/step - loss: 0.0887 - accuracy: 0.9739 - val_loss: 1.3307 - val_accuracy: 0.7098
Epoch 12/50
56/56 [==============================] - 579s 10s/step - loss: 0.1923 - accuracy: 0.9370 - val_loss: 1.3775 - val_accuracy: 0.7098
<keras.src.callbacks.History at 0x7e4235447280>
```

```
from tensorflow.keras import Sequential, layers
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers.experimental.preprocessing import Rescaling


full_model = Sequential()
full_model.add(Rescaling(1./255, input_shape=(150,150,3)))

full_model.add(layers.Conv2D(16, kernel_size=10, activation='relu'))
full_model.add(layers.MaxPooling2D(3))

full_model.add(layers.Conv2D(32, kernel_size=8, activation="relu"))
full_model.add(layers.MaxPooling2D(2))

full_model.add(layers.Conv2D(32, kernel_size=6, activation="relu"))
full_model.add(layers.MaxPooling2D(2))

full_model.add(layers.Flatten())
full_model.add(layers.Dense(50, activation='relu'))
full_model.add(layers.Dense(20, activation='relu'))
full_model.add(layers.Dense(5, activation='softmax'))


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


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

full_model.fit(train_ds, train_labels, epochs=50, validation_split=0.2, batch_size=32, callbacks=[es])
```

```
Epoch 1/50
56/56 [==============================] - 87s 2s/step - loss: 1.5687 - accuracy: 0.2601 - val_loss: 1.4823 - val_accuracy: 0.3605
Epoch 2/50
56/56 [==============================] - 87s 2s/step - loss: 1.4451 - accuracy: 0.3538 - val_loss: 1.4055 - val_accuracy: 0.4490
Epoch 3/50
56/56 [==============================] - 91s 2s/step - loss: 1.2653 - accuracy: 0.4560 - val_loss: 1.3450 - val_accuracy: 0.4490
Epoch 4/50
56/56 [==============================] - 91s 2s/step - loss: 1.2096 - accuracy: 0.4645 - val_loss: 1.2214 - val_accuracy: 0.4785
Epoch 5/50
56/56 [==============================] - 91s 2s/step - loss: 1.1372 - accuracy: 0.5168 - val_loss: 1.2819 - val_accuracy: 0.5193
Epoch 6/50
56/56 [==============================] - 92s 2s/step - loss: 1.0839 - accuracy: 0.5503 - val_loss: 1.1888 - val_accuracy: 0.5125
Epoch 7/50
56/56 [==============================] - 91s 2s/step - loss: 1.0196 - accuracy: 0.5866 - val_loss: 1.2190 - val_accuracy: 0.4853
Epoch 8/50
56/56 [==============================] - 92s 2s/step - loss: 0.9383 - accuracy: 0.6087 - val_loss: 1.1511 - val_accuracy: 0.5692
Epoch 9/50
56/56 [==============================] - 91s 2s/step - loss: 0.8711 - accuracy: 0.6388 - val_loss: 1.1964 - val_accuracy: 0.5193
Epoch 10/50
56/56 [==============================] - 87s 2s/step - loss: 0.8040 - accuracy: 0.6837 - val_loss: 1.2429 - val_accuracy: 0.5465
Epoch 11/50
56/56 [==============================] - 86s 2s/step - loss: 0.7200 - accuracy: 0.7161 - val_loss: 1.3619 - val_accuracy: 0.5329
Epoch 12/50
56/56 [==============================] - 86s 2s/step - loss: 0.6708 - accuracy: 0.7354 - val_loss: 1.3901 - val_accuracy: 0.5533
Epoch 13/50
56/56 [==============================] - 86s 2s/step - loss: 0.6612 - accuracy: 0.7411 - val_loss: 1.4603 - val_accuracy: 0.5465
<keras.src.callbacks.History at 0x7e421e972da0>
```