```
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[: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/tensorflo
Dl Completed...: 100%                                                5/5 [00:04<00:00,  1.09s/ file]
Dataset tf_flowers downloaded and prepared to /root/tensorflow_datasets/tf_flowers/3.0.1. Subsequent calls will reuse this data.
```

Image Preprocessing

```
## 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_ds[0].shape
```

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

```
## 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
```

```
## 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_kernel
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
```

```
## We will not train base model i.e. 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_layer (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688 (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
])
```

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

```
history=model.fit(train_ds, train_labels, epochs=10, validation_split=0.2, batch_size=32)
```

```
Epoch 1/5
65/65 ━━━━━━━━━━━━━━━━━━━━ 33s 274ms/step - accuracy: 0.3618 - loss: 2.6283 - val_accuracy: 0.5156 - val_loss: 1.1623
Epoch 2/5
65/65 ━━━━━━━━━━━━━━━━━━━━ 21s 174ms/step - accuracy: 0.5954 - loss: 1.0192 - val_accuracy: 0.6284 - val_loss: 1.0488
Epoch 3/5
65/65 ━━━━━━━━━━━━━━━━━━━━ 11s 168ms/step - accuracy: 0.7250 - loss: 0.7869 - val_accuracy: 0.6634 - val_loss: 1.0971
Epoch 4/5
65/65 ━━━━━━━━━━━━━━━━━━━━ 21s 179ms/step - accuracy: 0.7849 - loss: 0.5597 - val_accuracy: 0.6518 - val_loss: 0.9481
Epoch 5/5
65/65 ━━━━━━━━━━━━━━━━━━━━ 8s 127ms/step - accuracy: 0.8399 - loss: 0.4183 - val_accuracy: 0.7043 - val_loss: 0.9855
```

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

```
35/35 ━━━━━━━━━━━━━━━━━━━━ 13s 385ms/step - accuracy: 0.8811 - loss: 0.3352
Loss:  0.3205247223377228 Accuracy:  0.8864668607711792
```

```
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()
```