```python
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
)

train_ds[0].shape
```

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

```python
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], dtype=int64)>
```

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

train_labels[0]
```

```
<tf.Tensor: shape=(5,), dtype=float32, numpy=array([0., 0., 1., 0.,
0.], dtype=float32)>
```

## Use Pretrained VGG16 Image Classification model

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

train_ds[0].shape
```

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

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

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_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)

 Trainable params: 0 (0.00 B)

 Non-trainable 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
])

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 ━━━━━━━━━━━━━━━━━━━━ 1216s 19s/step - accuracy: 0.4054 - loss:
1.6197 - val_accuracy: 0.6031 - val_loss: 1.0109
Epoch 2/5
14/65 ━━━━━━━━━━━━━━━━━━━━ 3:06 4s/step - accuracy: 0.6267 - loss:
0.9620

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

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]
print(y_classes[:10])
print("\nTest")
print(test_labels[:10])

# Class names in the same order as dataset labels
class_names = ['dandelion', 'daisy', 'tulips', 'sunflowers', 'roses']

# Predict for one test image
y_pred_2 = model.predict(tf.expand_dims(test_ds[1], axis=0))
```

```python
# Show raw prediction probabilities
print("Predicted probabilities:", y_pred_2)

# Get the class index with highest probability
max_pred = np.argmax(y_pred_2)
print("Predicted class index:", max_pred)

# Print the corresponding class name
print("Predicted flower type:", class_names[max_pred])
```