```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.python.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
```

Preparing The Data

```
import pathlib
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, untar=True)
data_dir = pathlib.Path(data_dir)
```

```
print(data_dir)
```

```
/root/.keras/datasets/flower_photos
```

```
roses = list(data_dir.glob('roses/*'))
print(roses[0])
PIL.Image.open(str(roses[0]))
```

```
/root/.keras/datasets/flower_photos/roses/6879112993_5a29208438_n.jpg
```

```
img_height,img_width=180,180
batch_size=32
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
```

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

```
class_names = train_ds.class_names
print(class_names)
```

```
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(6):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Training The Model

```
resnet_model = Sequential()

pretrained_model= tf.keras.applications.ResNet50(include_top=False,
                   input_shape=(180,180,3),
                   pooling='avg',classes=5,
                   weights='imagenet')
for layer in pretrained_model.layers:
        layer.trainable=False

resnet_model.add(pretrained_model)
resnet_model.add(Flatten())
resnet_model.add(Dense(512, activation='relu'))
resnet_model.add(Dense(5, activation='softmax'))
```

```
resnet_model.summary()
```

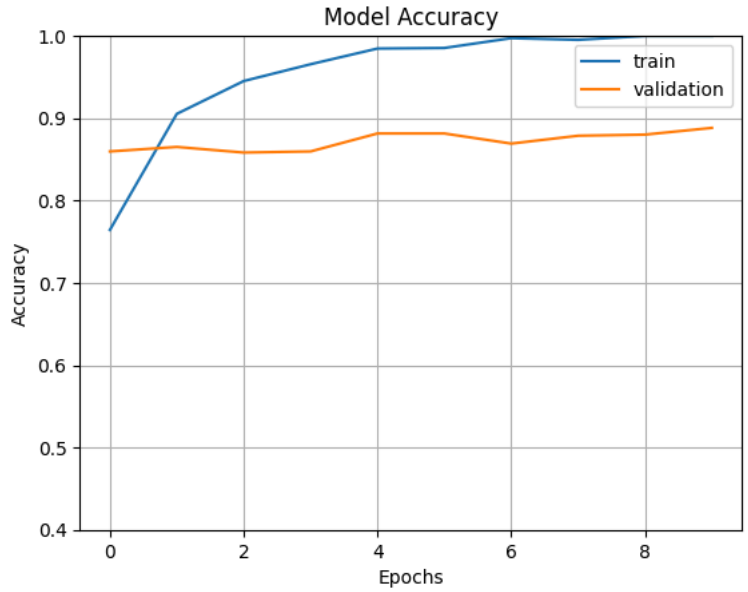
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 module_wrapper_3 (ModuleWr  (None, 2048)              0
 apper)

 module_wrapper_4 (ModuleWr  (None, 512)               1049088
 apper)

 module_wrapper_5 (ModuleWr  (None, 5)                 2565
 apper)

=================================================================
Total params: 24639365 (93.99 MB)
Trainable params: 1051653 (4.01 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
```

```
# One-hot encode the labels for multi-class classification
train_ds = train_ds.map(lambda x, y: (x, tf.one_hot(y, depth=5)))
val_ds = val_ds.map(lambda x, y: (x, tf.one_hot(y, depth=5)))

# Specify loss function as categorical_crossentropy
resnet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
history = resnet_model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

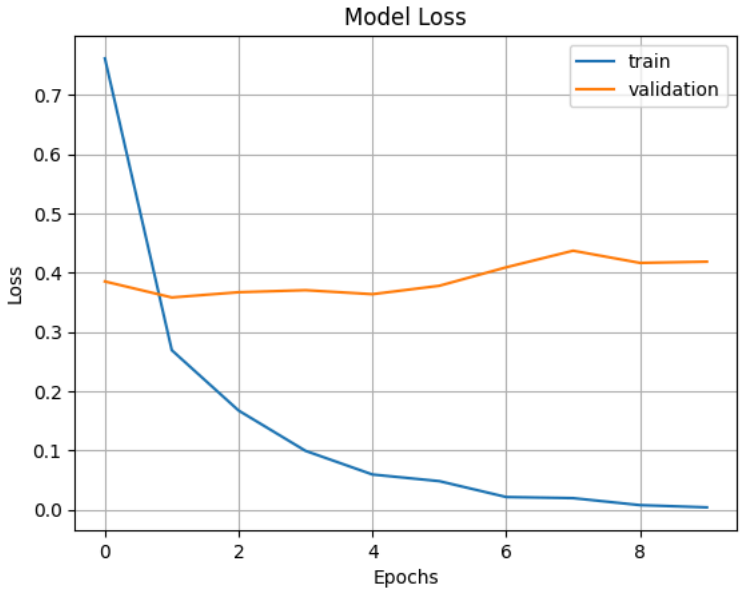
```
Epoch 1/10
92/92 [==============================] - 14s 110ms/step - loss: 0.7615 - accuracy: 0.7643 - val_loss: 0.3850 - val_accuracy: 0.8597
Epoch 2/10
92/92 [==============================] - 9s 99ms/step - loss: 0.2690 - accuracy: 0.9053 - val_loss: 0.3580 - val_accuracy: 0.8651
Epoch 3/10
92/92 [==============================] - 9s 90ms/step - loss: 0.1671 - accuracy: 0.9452 - val_loss: 0.3669 - val_accuracy: 0.8583
Epoch 4/10
92/92 [==============================] - 9s 88ms/step - loss: 0.0992 - accuracy: 0.9656 - val_loss: 0.3703 - val_accuracy: 0.8597
Epoch 5/10
92/92 [==============================] - 9s 94ms/step - loss: 0.0593 - accuracy: 0.9847 - val_loss: 0.3636 - val_accuracy: 0.8815
Epoch 6/10
92/92 [==============================] - 10s 101ms/step - loss: 0.0481 - accuracy: 0.9854 - val_loss: 0.3778 - val_accuracy: 0.8815
Epoch 7/10
92/92 [==============================] - 9s 97ms/step - loss: 0.0213 - accuracy: 0.9973 - val_loss: 0.4089 - val_accuracy: 0.8692
Epoch 8/10
92/92 [==============================] - 9s 92ms/step - loss: 0.0194 - accuracy: 0.9952 - val_loss: 0.4369 - val_accuracy: 0.8787
Epoch 9/10
92/92 [==============================] - 9s 93ms/step - loss: 0.0076 - accuracy: 1.0000 - val_loss: 0.4164 - val_accuracy: 0.8801
Epoch 10/10
92/92 [==============================] - 10s 101ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.4184 - val_accuracy: 0.8883
```

Evaluating The Model

```
fig1 = plt.gcf()
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.axis(ymin=0.4,ymax=1)
plt.grid()
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.grid()
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'validation'])
plt.show()
```



Making Predictions

```
import cv2
image=cv2.imread(str(roses[0]))
image_resized= cv2.resize(image, (img_height,img_width))
image=np.expand_dims(image_resized,axis=0)
print(image.shape)
```

```
(1, 180, 180, 3)
```

```
pred=resnet_model.predict(image)
print(pred)
```

```
1/1 [==============================] - 2s 2s/step
[[8.2993848e-11 5.9281069e-13 9.9999702e-01 5.1512104e-11 2.9831424e-06]]
```

```
output_class=class_names[np.argmax(pred)]
print("The predicted class is", output_class)
```

```
The predicted class is roses
```