```
!unzip "/content/Flowers-Dataset.zip"
```

```
  inflating: flowers/tulip/8712270243_8512c14bd.jpg
  inflating: flowers/tulip/8712270665_57b5bda0a2_n.jpg
  inflating: flowers/tulip/8712282563_3819afb7bc.jpg
  inflating: flowers/tulip/8713357842_9964a93473_n.jpg
  inflating: flowers/tulip/8713387500_6a9138b41b_n.jpg
  inflating: flowers/tulip/8713388322_e5ae26263b_n.jpg
  inflating: flowers/tulip/8713389178_66bceb71a8_n.jpg
  inflating: flowers/tulip/8713390684_041148dd3e_n.jpg
  inflating: flowers/tulip/8713391394_4b679ea1e3_n.jpg
  inflating: flowers/tulip/8713392604_90631fb809_n.jpg
  inflating: flowers/tulip/8713394070_b24561b0a9.jpg
  inflating: flowers/tulip/8713396140_5af8136136.jpg
  inflating: flowers/tulip/8713397358_0505cc0176_n.jpg
  inflating: flowers/tulip/8713397694_bcbcbba2c2_n.jpg
  inflating: flowers/tulip/8713398114_bc96f1b624_n.jpg
  inflating: flowers/tulip/8713398614_88202e452e_n.jpg
  inflating: flowers/tulip/8713398906_28e59a225a_n.jpg
  inflating: flowers/tulip/8713407768_f880df361f.jpg
  inflating: flowers/tulip/8717900362_2aa508e9e5.jpg
  inflating: flowers/tulip/8722514702_7ecc68691c.jpg
  inflating: flowers/tulip/8723767533_9145dec4bd_n.jpg
  inflating: flowers/tulip/8729501081_b993185542_m.jpg
  inflating: flowers/tulip/8733586143_3139db6e9e_n.jpg
  inflating: flowers/tulip/8748266132_5298a91dcf_n.jpg
  inflating: flowers/tulip/8750288831_5e49a9f29b.jpg
  inflating: flowers/tulip/8757486380_90952c5377.jpg
  inflating: flowers/tulip/8758464923_75a5ffe320_n.jpg
  inflating: flowers/tulip/8758519201_16e8d2d781_n.jpg
  inflating: flowers/tulip/8759594528_2534c0ec65_n.jpg
  inflating: flowers/tulip/8759597778_7fca5d434b_n.jpg
  inflating: flowers/tulip/8759601388_36e2a50d98_n.jpg
  inflating: flowers/tulip/8759606166_8e475013fa_n.jpg
  inflating: flowers/tulip/8759618746_f5e39fdbf8_n.jpg
  inflating: flowers/tulip/8762189906_8223cef62f.jpg
  inflating: flowers/tulip/8762193202_0fbf2f6a81.jpg
  inflating: flowers/tulip/8768645961_8f1e097170_n.jpg
  inflating: flowers/tulip/8817622133_a42bb90e38_n.jpg
  inflating: flowers/tulip/8838347159_746d14e6c1_m.jpg
  inflating: flowers/tulip/8838354855_c474fc66a3_m.jpg
  inflating: flowers/tulip/8838914676_8ef4db7f50_n.jpg
  inflating: flowers/tulip/8838975946_f54194894e_m.jpg
  inflating: flowers/tulip/8838983024_5c1a767878_n.jpg
  inflating: flowers/tulip/8892851067_79242a7362_n.jpg
  inflating: flowers/tulip/8904780994_8867d64155_n.jpg
  inflating: flowers/tulip/8908062479_449200a1b4.jpg
  inflating: flowers/tulip/8908097235_c3e746d36e_n.jpg
  inflating: flowers/tulip/9019694597_2d3bbedb17.jpg
  inflating: flowers/tulip/9030467406_05e93ff171_n.jpg
  inflating: flowers/tulip/9048307967_40a164a459_m.jpg
  inflating: flowers/tulip/924782410_94ed7913ca_m.jpg
  inflating: flowers/tulip/9378657435_89fabf13c9_n.jpg
  inflating: flowers/tulip/9444202147_405290415b_n.jpg
  inflating: flowers/tulip/9446982168_06c4d71da3_n.jpg
  inflating: flowers/tulip/9831362123_5aac525a99_n.jpg
  inflating: flowers/tulip/9870557734_88eb3b9e3b_n.jpg
  inflating: flowers/tulip/9947374414_fdf1d0861c_n.jpg
  inflating: flowers/tulip/9947385346_3a8cacea02_n.jpg
```

```
        inflating: flowers/tulip/9976515506_d496c5e72c.jpg
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt


batch_size = 16


data_augmentation = Sequential(
  [
    layers.RandomFlip("horizontal",input_shape=(180, 180, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)


train_data_set = tf.keras.utils.image_dataset_from_directory(
  "flowers",
  validation_split=0.25,
  subset="training",
  seed=132,
  image_size=(180, 180),
  batch_size=batch_size)
```

```
    Found 4317 files belonging to 5 classes.
    Using 3238 files for training.
```

```
val_data_set = tf.keras.utils.image_dataset_from_directory(
  "flowers",
  validation_split=0.25,
  subset="validation",
  seed=132,
  image_size=(180, 180),
  batch_size=batch_size)
```

```
    Found 4317 files belonging to 5 classes.
    Using 1079 files for validation.
```

```
class_names = train_data_set.class_names


plt.figure(figsize=(15, 15))
for images, labels in train_data_set.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]])
```


```
normalization_layer = layers.Rescaling(1./255)


dataset_normalized = train_data_set.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(dataset_normalized))
first_image = image_batch[0]
print(np.min(first_image), np.max(first_image))
```

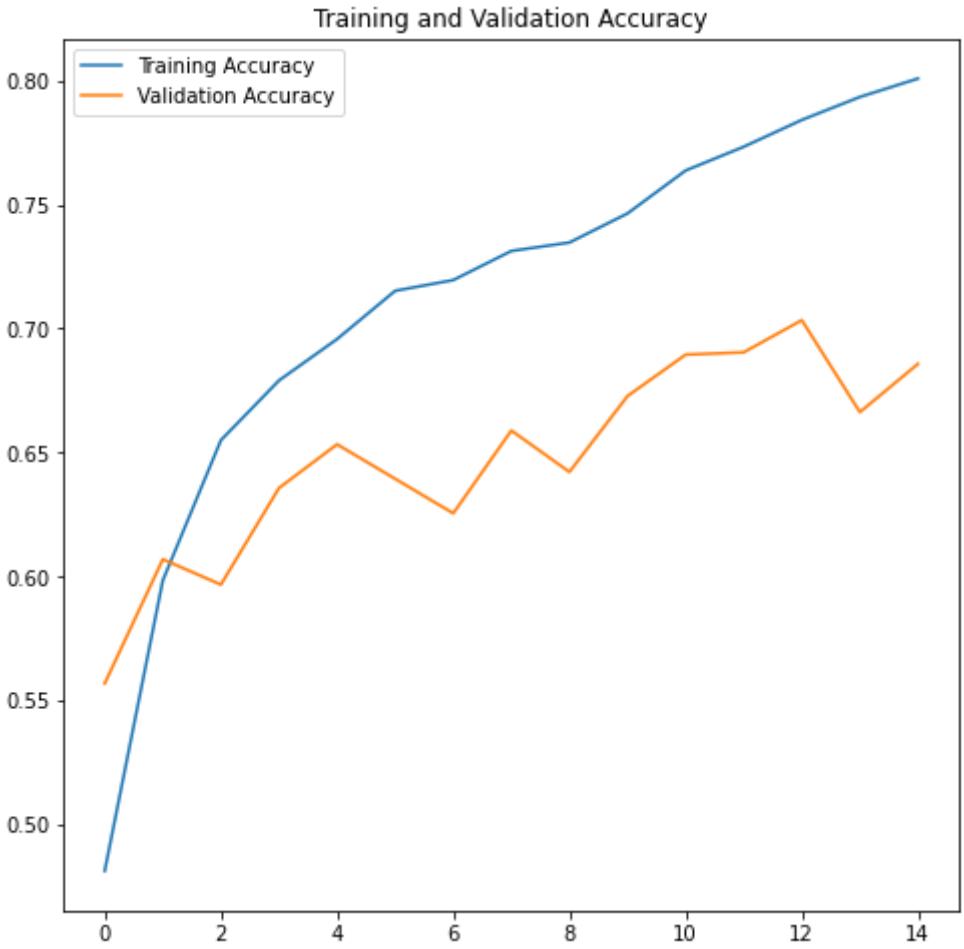
```
    0.0 1.0
```

```
num_classes = len(class_names)

model = Sequential([
  data_augmentation,
  layers.Rescaling(1./255, input_shape=(180, 180, 3)),
  # adding convolutional layer
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  # adding maxpooling layer
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  # adding flatten
  layers.Flatten(),
```

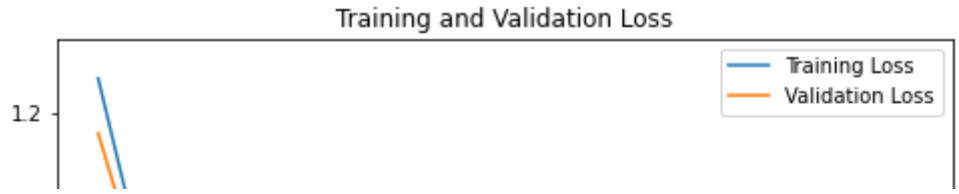
```
  # adding dense hidden layer
  layers.Dense(128, activation='relu'),
  # adding dense output layer
  layers.Dense(num_classes)
])


# compiling model with categorical cross entropy and adam optimizer
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])


epochs=15
history = model.fit(train_data_set,validation_data=val_data_set,epochs=epochs)
```

```
Epoch 1/15
203/203 [==============================] - 133s 633ms/step - loss: 1.2311 - accuracy
Epoch 2/15
203/203 [==============================] - 127s 623ms/step - loss: 1.0147 - accuracy
Epoch 3/15
203/203 [==============================] - 127s 624ms/step - loss: 0.9006 - accuracy
Epoch 4/15
203/203 [==============================] - 127s 623ms/step - loss: 0.8422 - accuracy
Epoch 5/15
203/203 [==============================] - 126s 622ms/step - loss: 0.8091 - accuracy
Epoch 6/15
203/203 [==============================] - 126s 621ms/step - loss: 0.7536 - accuracy
Epoch 7/15
203/203 [==============================] - 126s 620ms/step - loss: 0.7286 - accuracy
Epoch 8/15
203/203 [==============================] - 126s 621ms/step - loss: 0.7203 - accuracy
Epoch 9/15
203/203 [==============================] - 126s 619ms/step - loss: 0.7010 - accuracy
Epoch 10/15
203/203 [==============================] - 126s 620ms/step - loss: 0.6791 - accuracy
Epoch 11/15
203/203 [==============================] - 127s 623ms/step - loss: 0.6228 - accuracy
Epoch 12/15
203/203 [==============================] - 126s 621ms/step - loss: 0.6025 - accuracy
Epoch 13/15
203/203 [==============================] - 126s 621ms/step - loss: 0.5578 - accuracy
Epoch 14/15
203/203 [==============================] - 126s 622ms/step - loss: 0.5363 - accuracy
Epoch 15/15
203/203 [==============================] - 127s 623ms/step - loss: 0.5200 - accuracy
```

```
epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.plot(epochs_range, history.history['accuracy'], label='Training Accuracy')
plt.plot(epochs_range, history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```




```
plt.figure(figsize=(8, 8))
plt.plot(epochs_range, history.history['loss'], label='Training Loss')
plt.plot(epochs_range, history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Training and Validation Loss')
plt.show()
```




```
model.save("CNN Model for Classification Of Flowers.h5")
```

```
model.load_weights('CNN Model for Classification Of Flowers.h5')
```

```
sunflower_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/592
sunflower_path = tf.keras.utils.get_file('Red_sunflower', origin=sunflower_url)

img = tf.keras.utils.load_img(
    sunflower_path, target_size=(180, 180)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(class_names[np.argmax(score)],100 * np.max(score))
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example
117948/117948 [==============================] - 0s 0us/step
1/1 [==============================] - 0s 227ms/step
sunflower 96.24462127685547
```

Colab paid products - Cancel contracts here

✓ 0s completed at 9:26 PM