Student Name	R.Sai Nishit
Student Roll Number	2116190701179

Download the Dataset

```
[1]: import numpy as np
     import tensorflow as tf
     from tensorflow.keras import layers
     from tensorflow.keras.models import Sequential
     import matplotlib.pyplot as plt
     import os
```

```
[2]: batch_size = 16
```

Image Augmentation

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

```
[5]: os.listdir("C:\\Users\\Sai\\Flowers-Dataset")
```

```
t[5]: ['flowers']
```

```
[6]: train_data = tf.keras.utils.image_dataset_from_directory(
         "C:\\Users\\Marini\\Flowers-Dataset",
         validation_split=0.25,
         subset="training",
         seed=120,
         image_size=(180, 180),
         batch_size=batch_size)
     Found 4317 files belonging to 1 classes.
     Using 3238 files for training.
```

```
[7]: val_data_set = tf.keras.utils.image_dataset_from_directory(
         "C:\\Users\\Marini\\Flowers-Dataset",
         validation_split=0.25,
         subset="validation",
         seed=120,
         image_size=(180, 180),
         batch_size=batch_size)
     Found 4317 files belonging to 1 classes.
     Using 1079 files for validation.
```

```
[8]: class_names = train_data.class_names
```

```
[9]: plt.figure(figsize=(15, 15))
     for images, labels in train_data.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]])
```
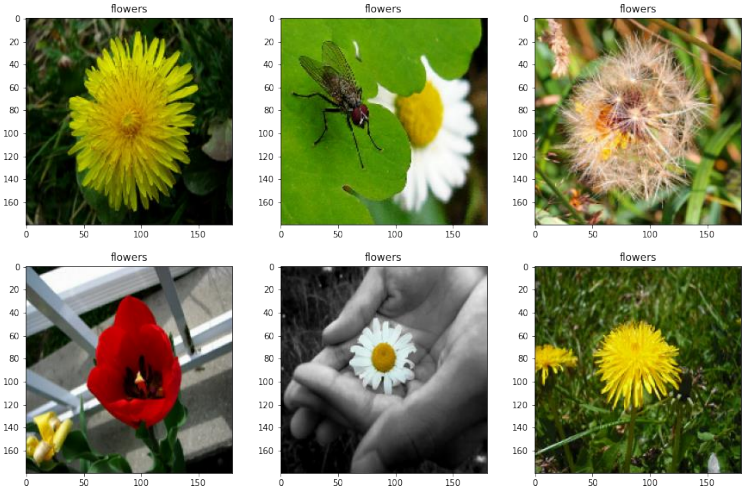


```
[10]: normalization_layer = layers.Rescaling(1./255)
```

```
[11]: dataset_normalized = train_data.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
```

Create Model

Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output)

```
[12]: num_classes = len(class_names)

      model = Sequential([
          data_aug,
          layers.Rescaling(1./255, input_shape=(180, 180, 3)),
          layers.Conv2D(16, 3, activation='relu'),
          layers.MaxPooling2D(), layers.Conv2D(32,
          3,activation='relu'), layers.Conv2D(32,
          3,activation='relu'),
          layers.MaxPooling2D(), layers.Conv2D(64,
          3, activation='relu'),
          layers.MaxPooling2D(), layers.Flatten(),
          layers.Dense(128, activation='relu'),
          layers.Dense(num_classes)
      ])
```

Compile The Model

compiling model with categorical cross entropy and adam optimizer

```
[13]: model.compile(optimizer='adam',
      loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      metrics=['accuracy'])
```
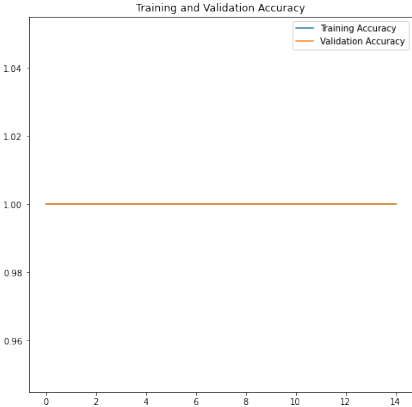
Fit The Model

```
[14]: epochs=15
      history = model.fit(train_data,validation_data=val_data_set,epochs=epochs)
      Epoch 1/15
      203/203 [==============================] - 33s 154ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 2/15
      203/203 [==============================] - 28s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 3/15
      203/203 [==============================] - 29s 143ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 4/15
      203/203 [==============================] - 29s 143ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 5/15
      203/203 [==============================] - 28s 139ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 6/15
      203/203 [==============================] - 29s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 7/15
      203/203 [==============================] - 28s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 8/15
      203/203 [==============================] - 29s 144ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 9/15
      203/203 [==============================] - 28s 137ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 10/15
      203/203 [==============================] - 28s 137ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 11/15
      203/203 [==============================] - 28s 139ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 12/15
      203/203 [==============================] - 29s 142ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 13/15
      203/203 [==============================] - 28s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 14/15
      203/203 [==============================] - 29s 143ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
      Epoch 15/15
      203/203 [==============================] - 29s 141ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

```
[15]: 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()
```



```
[16]: 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()
```
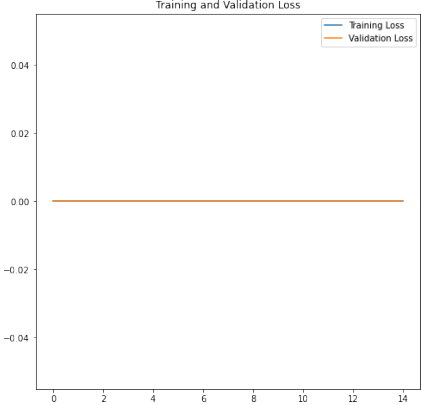

Save The Model

```
[17]: model.save("./flowers.h5")
```

```
[18]: model.load_weights("./flowers.h5")
```

Test The Model

```
[19]: from tensorflow.keras.preprocessing import image
      import numpy as np
```

```
img=image.load_img("C:\\Users\\Sai\\Flowers-Dataset\\flowers\\rose\\5172171681_5934378f08.jpg",target_size=(70,70))   img
```

[20]: