|  |  |
| --- | --- |
| **Student Name** | SASIDHARAN S |
| **Student Roll Number** | 190701193 |

# Download the Dataset

In[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

In[2]:

batch\_size**=** 16

# Image Augmentation

In[3]:

data\_aug**=** Sequential( [

layers**.**RandomFlip("horizontal",input\_shape**=**(180, 180, 3)), layers**.**RandomRotation(0.1),

layers**.**RandomZoom(0.1),

]

)

In[5]:

os**.**listdir("C:\\Users\\Harini\\Flowers-Dataset")

Out[5]:['flowers']

In[6]:

train\_data**=** tf**.**keras**.**utils**.**image\_dataset\_from\_directory( "C:\\Users\\Harini\\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.

In[7]:

val\_data\_set**=** tf**.**keras**.**utils**.**image\_dataset\_from\_directory( "C:\\Users\\Harini\\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.

In[8]:

class\_names**=** train\_data**.**class\_names

In[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]])



In[10]:

normalization\_layer**=** layers**.**Rescaling(1.**/**255)

In[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)

In[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

In[13]:

model**.**compile(optimizer**=**'adam', loss**=**tf**.**keras**.**losses**.**SparseCategoricalCrossentropy(from\_logits**=True**), metrics**=**['accuracy'])

# Fit The Model

In [14]:

epochs**=**15

history **=** model**.**fit(train\_data,validation\_data**=**val\_data\_set,epochs**=**epochs)

Epoch1/15

203/203 [==============================] - 33s 154ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch2/15

203/203 [==============================] - 28s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch3/15

203/203 [==============================] - 29s 141ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch4/15

203/203 [==============================] - 29s 143ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch5/15

203/203 [==============================] - 28s 139ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch6/15

203/203 [==============================] - 29s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch7/15

203/203 [==============================] - 28s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch8/15

203/203 [==============================] - 29s 144ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch9/15

203/203 [==============================] - 28s 137ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch10/15

203/203 [==============================] - 28s 137ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch11/15

203/203 [==============================] - 28s 139ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch12/15

203/203 [==============================] - 29s 142ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch13/15

203/203 [==============================] - 28s 140ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch14/15

203/203 [==============================] - 29s 143ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000 Epoch15/15

203/203 [==============================] - 29s 141ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val\_loss: 0.0000e+00 - val\_accuracy: 1.0000

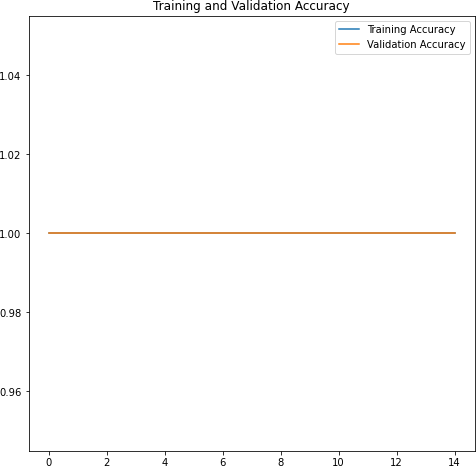
In[15]:

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()

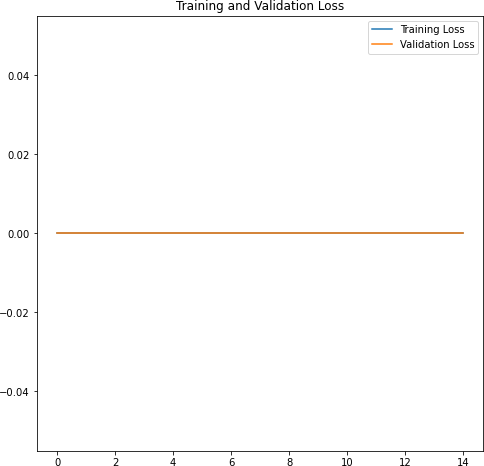


In[16]:

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

In[17]:

model**.**save("./flowers.h5")

In[18]:

model**.**load\_weights('./flowers.h5')

# Test The Model

In [19]:

**from** tensorflow.keras.preprocessing**import** image

**import** numpy**as** np

In[20]:

img**=**image**.**load\_img('C:\\Users\\Sai\\Flowers-Dataset\\flowers\\rose\\5172171681\_5934378f08.jpg',target\_size**=**(70,70)) img

Out[20]:

In[]: