import os  
import zipfile

Zip\_ref = zipfile.ZipFile("/content/drive/MyDrive/IBM/Assignment - 3/Flowers-Dataset.zip")  
Zip\_ref.extractall("/tmp")  
Zip\_ref.close()

import numpy as np  
import os  
import cv2  
import shutil  
import random as rn  
from tqdm import tqdm  
import matplotlib.pyplot as plt  
import tensorflow as tf  
from tensorflow import keras  
from tensorflow.keras import layers  
from tensorflow.keras.models import Sequential

data\_dir ="/tmp/flowers"

print(os.listdir("/tmp/flowers"))

['sunflower', 'daisy', 'tulip', 'rose', 'dandelion']

batch\_size = 32  
img\_height = 180  
img\_width = 180

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 4317 files belonging to 5 classes.  
Using 3454 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 4317 files belonging to 5 classes.  
Using 863 files for validation.

class\_names = train\_ds.class\_names  
print(class\_names)

['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']

import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10, 10))  
for images, labels in train\_ds.take(1):  
 for i in range(9):  
 ax = plt.subplot(3, 3, i + 1)  
 plt.imshow(images[i].numpy().astype("uint8"))  
 plt.title(class\_names[labels[i]])  
 plt.axis("off")



AUTOTUNE = tf.data.AUTOTUNE  
  
train\_ds = train\_ds.cache().shuffle(1000).prefetch(buffer\_size=AUTOTUNE)  
val\_ds = val\_ds.cache().prefetch(buffer\_size=AUTOTUNE)

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

normalized\_ds = train\_ds.map(lambda x, y: (normalization\_layer(x), y))  
image\_batch, labels\_batch = next(iter(normalized\_ds))  
first\_image = image\_batch[0]  
# Notice the pixels values are now in `[0,1]`.  
print(np.min(first\_image), np.max(first\_image))

0.0 1.0

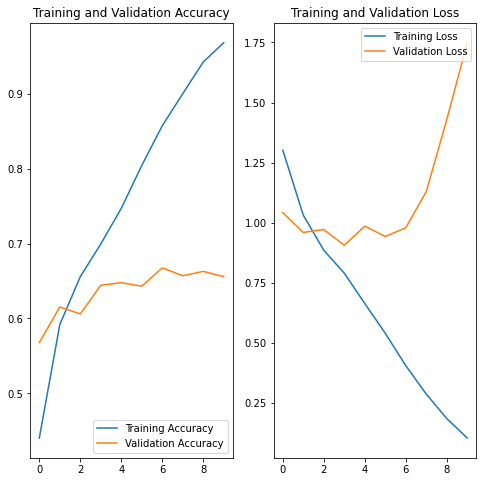
#Create model  
num\_classes = 5  
model = Sequential([  
 layers.experimental.preprocessing.Rescaling(1./255, input\_shape=(img\_height, img\_width, 3)),  
 layers.Conv2D(16, 3, padding='same', activation='relu'),  
 layers.MaxPooling2D(),  
 layers.Conv2D(32, 3, padding='same', activation='relu'),  
 layers.MaxPooling2D(),  
 layers.Conv2D(64, 3, padding='same', activation='relu'),  
 layers.MaxPooling2D(),  
 layers.Conv2D(128, 3, padding='same', activation='relu'),  
 layers.MaxPooling2D(),  
 layers.Flatten(),  
 layers.Dense(128, activation='relu'),  
 layers.Dense(num\_classes)  
])

#Compile & fit the model  
model.compile(optimizer='adam',  
 loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),  
 metrics=['accuracy'])

epochs=10  
history = model.fit(  
 train\_ds,  
 validation\_data=val\_ds,  
 epochs=epochs  
)

Epoch 1/10  
108/108 [==============================] - 96s 886ms/step - loss: 1.3016 - accuracy: 0.4404 - val\_loss: 1.0411 - val\_accuracy: 0.5678  
Epoch 2/10  
108/108 [==============================] - 94s 871ms/step - loss: 1.0308 - accuracy: 0.5918 - val\_loss: 0.9587 - val\_accuracy: 0.6153  
Epoch 3/10  
108/108 [==============================] - 93s 864ms/step - loss: 0.8844 - accuracy: 0.6558 - val\_loss: 0.9710 - val\_accuracy: 0.6060  
Epoch 4/10  
108/108 [==============================] - 103s 956ms/step - loss: 0.7890 - accuracy: 0.6995 - val\_loss: 0.9051 - val\_accuracy: 0.6443  
Epoch 5/10  
108/108 [==============================] - 95s 878ms/step - loss: 0.6627 - accuracy: 0.7470 - val\_loss: 0.9848 - val\_accuracy: 0.6477  
Epoch 6/10  
108/108 [==============================] - 94s 873ms/step - loss: 0.5386 - accuracy: 0.8043 - val\_loss: 0.9419 - val\_accuracy: 0.6431  
Epoch 7/10  
108/108 [==============================] - 93s 866ms/step - loss: 0.4038 - accuracy: 0.8573 - val\_loss: 0.9779 - val\_accuracy: 0.6674  
Epoch 8/10  
108/108 [==============================] - 94s 872ms/step - loss: 0.2854 - accuracy: 0.9001 - val\_loss: 1.1288 - val\_accuracy: 0.6570  
Epoch 9/10  
108/108 [==============================] - 94s 870ms/step - loss: 0.1834 - accuracy: 0.9424 - val\_loss: 1.4286 - val\_accuracy: 0.6628  
Epoch 10/10  
108/108 [==============================] - 94s 873ms/step - loss: 0.1022 - accuracy: 0.9682 - val\_loss: 1.7492 - val\_accuracy: 0.6559

#Test the model to know the results  
acc = history.history['accuracy']  
val\_acc = history.history['val\_accuracy']  
  
loss = history.history['loss']  
val\_loss = history.history['val\_loss']  
  
epochs\_range = range(epochs)  
  
plt.figure(figsize=(8, 8))  
plt.subplot(1, 2, 1)  
plt.plot(epochs\_range, acc, label='Training Accuracy')  
plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')  
plt.legend(loc='lower right')  
plt.title('Training and Validation Accuracy')  
  
plt.subplot(1, 2, 2)  
plt.plot(epochs\_range, loss, label='Training Loss')  
plt.plot(epochs\_range, val\_loss, label='Validation Loss')  
plt.legend(loc='upper right')  
plt.title('Training and Validation Loss')  
plt.show()



#Image Augmentation  
data\_augmentation = keras.Sequential(  
 [  
 layers.experimental.preprocessing.RandomFlip("horizontal",   
 input\_shape=(img\_height,   
 img\_width,  
 3)),  
 layers.experimental.preprocessing.RandomRotation(0.1),  
 layers.experimental.preprocessing.RandomZoom(0.1),  
 ]  
)

plt.figure(figsize=(10, 10))  
for images, \_ in train\_ds.take(1):  
 for i in range(9):  
 augmented\_images = data\_augmentation(images)  
 ax = plt.subplot(3, 3, i + 1)  
 plt.imshow(augmented\_images[0].numpy().astype("uint8"))  
 plt.axis("off")



#Save the model  
model.save('flowers\_model2.h5')

from tensorflow.keras.models import load\_model  
model2 = load\_model('flowers\_model2.h5')