Assignment-3
BuildCNNmodelforclassificationofFlowers

AssignmentDate	03October2022
TeamID	PNT2022TMID00678
ProjectName	AIBASEDDISCOURSEFORBANKINGINDUSTRY
StudentName	ISWARYA G
StudentRoll Number	211419104501
MaximumMarks	2Marks

Question-1. Loadthedataset

Solution:

!unzipFlowers-Dataset.zip

```
inflating: flowers/daisy/1396526833_fb867165be_n.jpg
inflating: flowers/daisy/13977181862_f8237b6b52.jpg
inflating: flowers/daisy/14021430525_e06baf93a9.jpg
inflating: flowers/daisy/14073784469_ffb12f3387_n.jpg
inflating: flowers/daisy/14087947408_9779257411_n.jpg
inflating: flowers/daisy/14088053307_1a13a0bf91_n.jpg
inflating: flowers/daisy/14114116486_0bb6649bc1_m.jpg
inflating: flowers/daisy/14147016029_8d3cf2414e.jpg
inflating: flowers/daisy/14163875973_467224aaf5_m.jpg
inflating: flowers/daisy/14167534527_781ceb1b7a_n.jpg
inflating: flowers/daisy/14167543177_cd36b54ac6_n.jpg
inflating: flowers/daisy/14219214466_3ca6104eae_m.jpg
inflating: flowers/daisy/14221836990_90374e6b34.jpg
inflating: flowers/daisy/14221848160_7f0a37c395.jpg
inflating: flowers/daisy/14245834619_153624f836.jpg
inflating: flowers/daisy/14264136211_9531fbc144.jpg
inflating: flowers/daisy/14272874304_47c0a46f5a.jpg
inflating: flowers/daisy/14307766919_fac3c37a6b_m.jpg
inflating: flowers/daisy/14330343061_99478302d4_m.jpg
inflating: flowers/daisy/14332947164_9b13513c71_m.jpg
inflating: flowers/daisy/14333681205_a07c9f1752_m.jpg
inflating: flowers/daisy/14350958832_29bdd3a254.jpg
inflating: flowers/daisy/14354051035_1037b30421_n.jpg
inflating: flowers/daisy/14372713423_61e2daae88.jpg
inflating: flowers/daisy/14399435971_ea5868c792.jpg
inflating: flowers/daisy/14402451388_56545a374a_n.jpg
inflating: flowers/daisy/144076848_57e1d662e3_m.jpg
```

```
inflating: flowers/daisy/14372713423_61e2daae88.jpg
inflating: flowers/daisy/14399435971_ea5868c792.jpg
inflating: flowers/daisy/14402451388_56545a374a_n.jpg
inflating: flowers/daisy/144076848_57e1d662e3_m.jpg
inflating: flowers/daisy/144099102_bf63a41e4f_n.jpg
inflating: flowers/daisy/1441939151_b271408c8d_n.jpg
inflating: flowers/daisy/14421389519_d5fd353eb4.jpg
inflating: flowers/daisy/144603918_b9de002f60_m.jpg
inflating: flowers/daisy/14471433500_cdaa22e3ea_m.jpg
inflating: flowers/daisy/14485782498_fb342ec301.jpg
inflating: flowers/daisy/14507818175_05219b051c_m.jpg
inflating: flowers/daisy/14523675369_97c31d0b5b.jpg
inflating: flowers/daisy/14551098743_2842e7a004_n.jpg
inflating: flowers/daisy/14554906452_35f066ffe9_n.jpg
inflating: flowers/daisy/14564545365_1f1d267bf1_n.jpg
inflating: flowers/daisy/14569895116_32f0dcb0f9.jpg
inflating: flowers/daisy/14591326135_930703dbed_m.jpg
inflating: flowers/daisy/14600779226_7bbc288d40_m.jpg
inflating: flowers/daisy/14613443462_d4ed356201.jpg
inflating: flowers/daisy/14621687774_ec52811acd_n.jpg
inflating: flowers/daisy/14674743211_f68b13f6d9.jpg
inflating: flowers/daisy/14698531521_0c2f0c6539.jpg
inflating: flowers/daisy/147068564_32bb4350cc.jpg
inflating: flowers/daisy/14707111433_cce08ee007.jpg
inflating: flowers/daisy/14716799982_ed6d626a66.jpg
inflating: flowers/daisy/14816364517_2423021484_m.jpg
inflating: flowers/daisy/14866200659_6462c723cb_m.jpg
```

```python
#importing required libraries to build a CNN classification model with accuracy

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 = 32
img_height = 180
img_width = 180
data_dir = "/content/flowers"
```

Question-2. ImageAugmentation

Solution:

fromtensorflow.keras.preprocessing.image importImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_flip = True, zoom_range=0.2)

x_train=train_datagen.flow_from_directory(r"/content/flowers",target_size=(64,64),class_mode ="categorical", batch_size=100)

```
Found 4317 images belonging to 5 classes.
```

```
#Image Augumentation accuracy
data_augmentation = Sequential(
  [
    layers.RandomFlip("horizontal",input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)
```

Question-3. Createmodel -ModelBuildingandalsoSplitdatasetintotrainingandtestingsets

Solution:

fromtensorflow.keras.modelsimportSequential

from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Densemodel=Sequential()

**train_ds =
tf.keras.utils.image_dataset_from_directory(data_dir,
validation_split=0.2,
subset="training",se
ed=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.utils.image_dataset_from_directory(data_di
 r,
 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_namesprint(class_na
mes)
```

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

```
plt.figure(figsize=(10,10))
for images, labels in
 train_ds.take(1):foriinrange(9):
  ax = plt.subplot(3, 3, i +
  1)plt.imshow(images[i].numpy().astype("uint8"))plt.ti
  tle(class_names[labels[i]])
  plt.axis("off")
```

Question-4. Addthelayers(Convolution,MaxPooling,Flatten,Dense-(HiddenLayers),Output)

Solution:

model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3)

))model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())model.add(Dense(300

,activation="relu"))

model.add(Dense(150, activation = "relu")) #mulitple dense

layersmodel.add(Dense(5,activation="softmax"))#outputlayer

```
#Adding the layers for accuracy
num_classes = len(class_names)

model = Sequential([
  data_augmentation,
  layers.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.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Question-5. CompileTheModel

Solution:

model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")len(x_train)

```
44
```

#Compilethemodelforfurtheraccuracy

```
model.compile(optimizer='adam',loss=tf.keras.losses.SparseCategoricalCrossentr
        opy(from_logits=True),metrics=['accuracy'])
epochs=10
history =
 model.fit(train_ds,valid
 ation_data=val_ds,epoc
 hs=epochs
)
```

```
Epoch 1/10
108/108 [==============================] - 132s 1s/step - loss: 1.2821 - accuracy: 0.4537 - val_loss: 1.0988 - val_accuracy: 0.5458
Epoch 2/10
108/108 [==============================] - 130s 1s/step - loss: 1.0298 - accuracy: 0.5921 - val_loss: 0.9494 - val_accuracy: 0.6304
Epoch 3/10
108/108 [==============================] - 129s 1s/step - loss: 0.9274 - accuracy: 0.6448 - val_loss: 0.9927 - val_accuracy: 0.6130
Epoch 4/10
108/108 [==============================] - 129s 1s/step - loss: 0.9000 - accuracy: 0.6642 - val_loss: 0.9264 - val_accuracy: 0.6419
Epoch 5/10
108/108 [==============================] - 136s 1s/step - loss: 0.8432 - accuracy: 0.6778 - val_loss: 0.8499 - val_accuracy: 0.6674
Epoch 6/10
108/108 [==============================] - 130s 1s/step - loss: 0.8166 - accuracy: 0.6888 - val_loss: 0.8714 - val_accuracy: 0.6732
Epoch 7/10
108/108 [==============================] - 130s 1s/step - loss: 0.7726 - accuracy: 0.7064 - val_loss: 0.7873 - val_accuracy: 0.6895
Epoch 8/10
108/108 [==============================] - 130s 1s/step - loss: 0.7262 - accuracy: 0.7250 - val_loss: 0.7957 - val_accuracy: 0.6860
Epoch 9/10
108/108 [==============================] - 128s 1s/step - loss: 0.7094 - accuracy: 0.7284 - val_loss: 0.7960 - val_accuracy: 0.7068
Epoch 10/10
108/108 [==============================] - 130s 1s/step - loss: 0.6820 - accuracy: 0.7383 - val_loss: 0.7914 - val_accuracy: 0.6941
```

#TofindtheTrainingandValidation-Accuracy&Loss(Visualization)

acc=history.history['accuracy']
val_acc=history.history['val_accuracy']

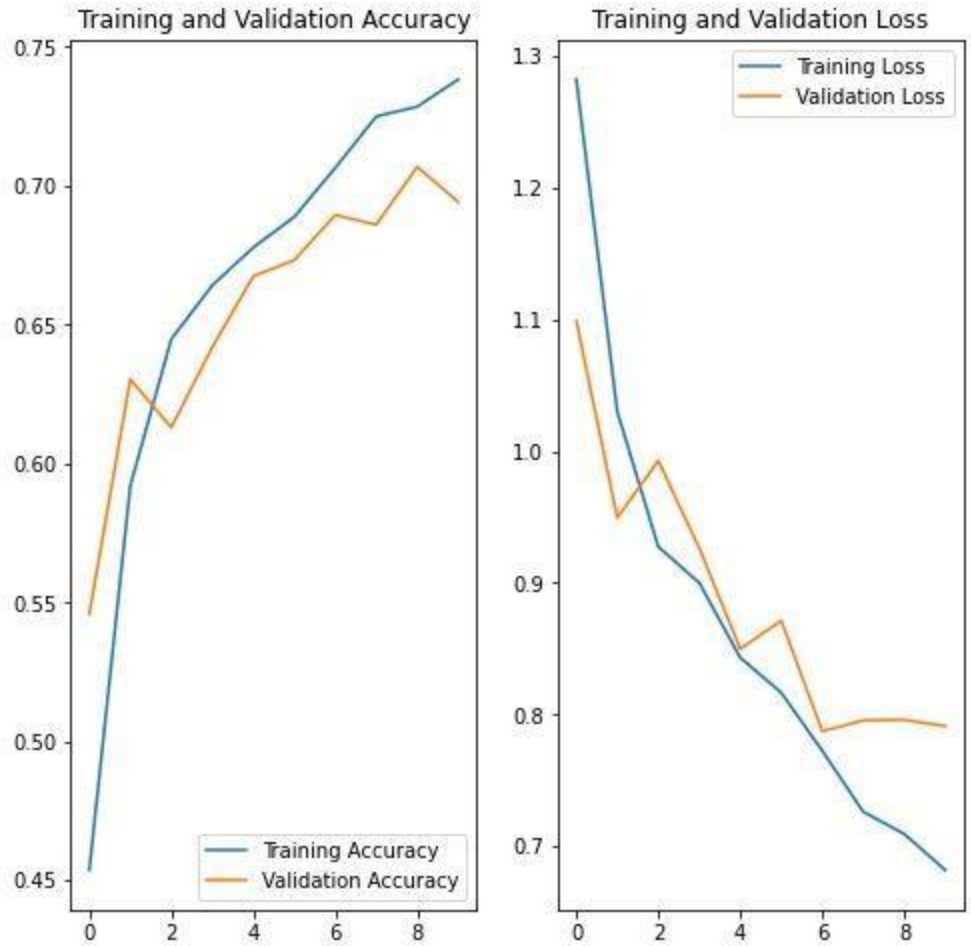
loss=history.history['loss']val_loss=
history.history['val_loss']

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='lowerright')
plt.title('TrainingandValidationAccuracy')

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='upperright')
plt.title('Training and Validation
Loss')plt.show()



Question-6. FitTheModel

Solution:

model.fit(x_train,epochs=15,steps_per_epoch=len(x_train))

```
Epoch 1/15
44/44 [==============================] - 31s 684ms/step - loss: 1.7914 - accuracy: 0.3588
Epoch 2/15
44/44 [==============================] - 29s 648ms/step - loss: 1.1730 - accuracy: 0.5045
Epoch 3/15
44/44 [==============================] - 29s 650ms/step - loss: 1.0967 - accuracy: 0.5529
Epoch 4/15
44/44 [==============================] - 29s 648ms/step - loss: 1.0351 - accuracy: 0.5939
Epoch 5/15
44/44 [==============================] - 29s 645ms/step - loss: 0.9920 - accuracy: 0.6127
Epoch 6/15
44/44 [==============================] - 30s 677ms/step - loss: 0.9659 - accuracy: 0.6259
Epoch 7/15
44/44 [==============================] - 29s 648ms/step - loss: 0.9129 - accuracy: 0.6426
Epoch 8/15
44/44 [==============================] - 29s 647ms/step - loss: 0.9085 - accuracy: 0.6433
Epoch 9/15
44/44 [==============================] - 32s 717ms/step - loss: 0.8597 - accuracy: 0.6620
Epoch 10/15
44/44 [==============================] - 30s 674ms/step - loss: 0.8350 - accuracy: 0.6824
Epoch 11/15
44/44 [==============================] - 29s 648ms/step - loss: 0.8420 - accuracy: 0.6718
Epoch 12/15
44/44 [==============================] - 29s 650ms/step - loss: 0.7857 - accuracy: 0.7030
Epoch 13/15
44/44 [==============================] - 29s 649ms/step - loss: 0.7868 - accuracy: 0.7000
Epoch 14/15
44/44 [==============================] - 29s 650ms/step - loss: 0.7542 - accuracy: 0.7132
Epoch 15/15
44/44 [==============================] - 30s 676ms/step - loss: 0.7467 - accuracy: 0.7107
<keras.callbacks.History at 0x7f602ce90090>
```

Question-7. SaveTheModel

Solution:

model.save("flowers.h1")

model.save("flowers.m5")#anothermodeltoshowtheaccuracy

Question-8. TestTheModel

Solution:

from tensorflow.keras.models import

load_modelfrom tensorflow.keras.preprocessing

import imageimportnumpy asnp

model=load_model("/content/flowers.h1")

#Testingwitharandomroseimagefromgoogle

img=image.load_img("/content/rose.gif",target_size=(64,64))

img

x =

image.img_to_array(img)x.n

3

dim

x = np.expand_dims(x,axis =

0)x.ndim

4

pred =

model.predict(x)pred

```
array([[0., 0., 1., 0., 0.]], dtype=float32)
```

labels=['daisy','dandelion','roses','sunflowers','tulips']

labels[np.argmax(pred)]

```
'roses'
```

#Testingthealternativemodelwithaccuracy

```
sunflower_url="https://storage.googleapis.com/download.tensorflow.org/example_images/592p
x-Red_sunflower.jpg"

sunflower_path = tf.keras.utils.get_file('Red_sunflower',

origin=sunflower_url)img=tf.keras.utils.load_img(

   sunflower_path,target_size=(img_height,img_width)

)

img_array=tf.keras.utils.img_to_array(img)

img_array=tf.expand_dims(img_array, 0)#Createabatch

predictions =

model.predict(img_array)score =

tf.nn.softmax(predictions[0])print(

   "Thisimagemostlikelybelongsto{}witha{:.2f}percentconfidence."

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

)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/592px-Red_sunflower.jpg
122880/117948 [===============================] - 0s 0us/step
131072/117948 [================================] - 0s 0us/step
This image most likely belongs to sunflower with a 99.85 percent confidence.
```