Assignment -3
Build CNN Model for Classification of Flowers

Assignment Date	13 October 2022
Student Name	ABISHEK GOVINTHAN K B
Student Roll Number	912819104005
Project	ARRHYTHMIA BY USING DEEP LEARNING WITH 2-D ECG SPECTRAL IMAGE REPRESENTATION
Maximum Marks	2 Marks

Question-1:

Download the Dataset: Dataset

Solution:

from google.colab import drive
drive.mount('/content/drive')

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

!unzip /content/drive/MyDrive/Flowers-Dataset.zip

```
!unzip /content/drive/MyDrive/Flowers-Dataset.zip
```

```
Archive:  /content/drive/MyDrive/Flowers-Dataset.zip
  inflating: flowers/daisy/100080576_f52e8ee070_n.jpg
  inflating: flowers/daisy/10140303196_b88d3d6cec.jpg
  inflating: flowers/daisy/10172379554_b296050f82_n.jpg
  inflating: flowers/daisy/10172567486_2748826a8b.jpg
  inflating: flowers/daisy/10172636503_21bededa75_n.jpg
  inflating: flowers/daisy/102841525_bd6628ae3c.jpg
  inflating: flowers/daisy/10300722094_28fa978807_n.jpg
  inflating: flowers/daisy/1031799732_e7f4008c03.jpg
  inflating: flowers/daisy/10391248763_1d16681106_n.jpg
  inflating: flowers/daisy/10437754174_22ec990b77_m.jpg
  inflating: flowers/daisy/10437770546_8bb6f7bdd3_m.jpg
  inflating: flowers/daisy/10437929963_bc13eebe0c.jpg
  inflating: flowers/daisy/10466290366_cc72e33532.jpg
  inflating: flowers/daisy/10466558316_a7198b87e2.jpg
  inflating: flowers/daisy/11124324295_503f3a0804.jpg
  inflating: flowers/daisy/1140299375_3aa7024466.jpg
  inflating: flowers/daisy/11439894966_dca877f0cd.jpg
  inflating: flowers/daisy/1150395827_6f94a5c6e4_n.jpg
  inflating: flowers/daisy/11642632_1e7627a2cc.jpg
  inflating: flowers/daisy/11834945233_a53b7a92ac_m.jpg
  inflating: flowers/daisy/11870378973_2ec1919f12.jpg
  inflating: flowers/daisy/11891885265_ccefec7284_n.jpg
  inflating: flowers/daisy/12193032636_b50ae7db35_n.jpg
  inflating: flowers/daisy/12348343085_d4c396e5b5_m.jpg
  inflating: flowers/daisy/12585131704_0f64b17059_m.jpg
  inflating: flowers/daisy/12601254324_3cb62c254a_m.jpg
  inflating: flowers/daisy/1265350143_6e2b276ec9.jpg
  inflating: flowers/daisy/12701063955_4840594ea6_n.jpg
  inflating: flowers/daisy/1285423653_18926dc2c8_n.jpg
  inflating: flowers/daisy/1286274236_1d7ac84efb_n.jpg
  inflating: flowers/daisy/12891819633_e4c82b51e8.jpg
  inflating: flowers/daisy/1299501272_59d9da5510_n.jpg
  inflating: flowers/daisy/1306119996_ab8ae14d72_n.jpg
  inflating: flowers/daisy/1314069875_da8dc023c6_m.jpg
  inflating: flowers/daisy/1342002397_9503c97b49.jpg
  inflating: flowers/daisy/134409839_71069a95d1_m.jpg
  inflating: flowers/daisy/1344985627_c3115e2d71_n.jpg
  inflating: flowers/daisy/13491959645_2cd9df44d6_n.jpg
  inflating: flowers/daisy/1354396826_2868631432_m.jpg
  inflating: flowers/daisy/1355787476_32e9f2a30b.jpg
  inflating: flowers/daisy/13583238844_573df2de8e_m.jpg
  inflating: flowers/daisy/1374193928_a52320eafa.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 = 32
img_height = 180
img_width = 180
data_dir = "/content/flowers"

```
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:

Image Augmentation

Solution:

from tensorflow.keras.preprocessing.image import ImageDataGenerator
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/drive/MyDrive/flowers", target_size = (64,64) , class_mode = "categorical", batch_size = 100)

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
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/drive/MyDrive/flowers", target_size = (64,64) , class_mode = "categorical", batch_size = 100)
```

```
Found 4355 images belonging to 5 classes.
```

Question-3:

Create Model

Solution:

from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense model = Sequential()

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

```
model = Sequential()
```

```
import tensorflow as tf
train_ds =
 tf.keras.utils.image_dataset_from_directory( data_dir,
 validation_split=0.2,
 subset="training",
 seed=123,
 image_size=(img_height, img_width),
 batch_size=batch_size)
```

```
import tensorflow as tf
train_ds = tf.keras.utils.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.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
val_ds = tf.keras.utils.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
class_names
```

```
class_names = train_ds.class_names
class_names
```

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

plt.figure(figsize=(5, 5))
for images, labels in train_ds.take(1):
for i in range(6):
ax = plt.subplot(2, 3, i + 1)
plt.imshow(images[i].numpy().astype("uint8"))
plt.title(class_names[labels[i]])
plt.axis("off")

```
plt.figure(figsize=(5, 5))
for images, labels in train_ds.take(1):
  for i in range(6):
    ax = plt.subplot(2, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Question-4:

Add Layers (Convolution, Max Pooling, Flatten, Dense-(Hidden Layers),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
layers model.add(Dense(5, activation = "softmax")) #output layer

```
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 layers
model.add(Dense(5, activation = "softmax")) #output layer
```

Question-5:

Compile The Model

Solution:

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

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

```
44
```

Question-6:

Fit The Model

Solution:

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

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

```
Epoch 1/15
44/44 [==============================] - 637s 15s/step - loss: 1.1897 - accuracy: 0.4907
Epoch 2/15
44/44 [==============================] - 35s 797ms/step - loss: 1.0953 - accuracy: 0.5635
Epoch 3/15
44/44 [==============================] - 35s 781ms/step - loss: 1.0546 - accuracy: 0.5800
Epoch 4/15
44/44 [==============================] - 36s 821ms/step - loss: 0.9853 - accuracy: 0.6216
Epoch 5/15
44/44 [==============================] - 35s 797ms/step - loss: 0.9397 - accuracy: 0.6342
Epoch 6/15
44/44 [==============================] - 35s 787ms/step - loss: 0.9066 - accuracy: 0.6443
Epoch 7/15
44/44 [==============================] - 35s 779ms/step - loss: 0.8930 - accuracy: 0.6542
Epoch 8/15
44/44 [==============================] - 34s 779ms/step - loss: 0.8492 - accuracy: 0.6668
Epoch 9/15
44/44 [==============================] - 35s 787ms/step - loss: 0.8074 - accuracy: 0.6877
Epoch 10/15
44/44 [==============================] - 35s 785ms/step - loss: 0.8093 - accuracy: 0.6845
Epoch 11/15
44/44 [==============================] - 35s 799ms/step - loss: 0.7935 - accuracy: 0.6948
Epoch 12/15
44/44 [==============================] - 35s 785ms/step - loss: 0.7712 - accuracy: 0.7063
Epoch 13/15
44/44 [==============================] - 36s 819ms/step - loss: 0.7506 - accuracy: 0.7098
Epoch 14/15
44/44 [==============================] - 35s 802ms/step - loss: 0.7350 - accuracy: 0.7187
Epoch 15/15
44/44 [==============================] - 35s 788ms/step - loss: 0.7238 - accuracy: 0.7235
<keras.callbacks.History at 0x7f579ebddad0>
```

Question-7:

Save The Model

Solution:

model.save("flowers.h5")
model.save("flowers.m5")#another model

```
model.save("flowers.h5")
```

```
model.save("flowers.m5")#another model to show the accuracy
```

Question-8:

Test The Model

Solution:

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
model = load_model("/content/flowers.h5")
img = image.load_img("/content/drive/MyDrive/rose.gif", target_size = (64,64))
img

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
model = load_model("/content/flowers.h5")
```

```
#Testing with a random rose image from Google

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

x = image.img_to_array(img)
x.ndim

```
x = image.img_to_array(img)
x.ndim
```

```
3
```

x = np.expand_dims(x,axis =
0) x.ndim

```
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'
```

```
sunflower_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/592px-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) # Create a batch

pred
score = tf.nn.softmax(pred[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score)))
```

```
sunflower_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/592px-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) # Create a batch

pred
score = tf.nn.softmax(pred[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to rose with a 40.46 percent confidence.
```