Assignment -3

Build CNN Model for Classification of Flowers

Assignment Date	:	06-10-2022
Student Name	:	P.Harini Brindha
Student Roll Number	:	912419104008
Project	:	Fertilizer Recommendation System for Disease Prediction
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')
```

```
Mounted at /content/drive
```

!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/10555749515_13a12a026e.jpg
  inflating: flowers/daisy/10555815624_dc211569b0.jpg
  inflating: flowers/daisy/10555826524_423eb8bf71_n.jpg
  inflating: flowers/daisy/10559679065_50d2b16f6d.jpg
  inflating: flowers/daisy/105806915_a9c13e2106_n.jpg
  inflating: flowers/daisy/10712722853_5632165b04.jpg
  inflating: flowers/daisy/107592979_aaa9cdfe78_m.jpg
  inflating: flowers/daisy/10770585085_4742b9dac3_n.jpg
  inflating: flowers/daisy/10841136265_af473efc60.jpg
  inflating: flowers/daisy/10993710036_2033222c91.jpg
  inflating: flowers/daisy/10993818044_4c19b86c82.jpg
  inflating: flowers/daisy/10994032453_ac7f8d9e2e.jpg
  inflating: flowers/daisy/11023214096_b5b39fab08.jpg
  inflating: flowers/daisy/11023272144_fce94401f2_m.jpg
  inflating: flowers/daisy/11023277956_8980d53169_m.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
```

Question-2:

Image Agumentation

Solution:

from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_fli
p = True, zoom_range = 0.2)
test_datagen = ImageDataGenerator(rescale= 1./255)
x_train = train_datagen.flow_from_directory(r"/content/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)
```

```
test_datagen = ImageDataGenerator(rescale= 1./255)
```

```
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.
```

Question-3:

Create Model

Solution:

from tensorflow.keras.models import Sequential
model = Sequential()

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

```
model = Sequential()
```

Question-4:

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

Solution:

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

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

```
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 = "a
dam")
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, validation_data = x_test, steps_per_epoch = len(x_train),
validation_steps = len(x_test))
model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))

```
#model.fit(x_train, epochs = 15, validation_data = x_test, steps_per_epoch = len(x_train), validation_steps = len(x_test))
```

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

```
Epoch 1/15
44/44 [==============================] - 30s 645ms/step - loss: 1.7030 - accuracy: 0.3584
Epoch 2/15
44/44 [==============================] - 28s 654ms/step - loss: 1.1869 - accuracy: 0.5073
Epoch 3/15
44/44 [==============================] - 29s 643ms/step - loss: 1.0883 - accuracy: 0.5643
Epoch 4/15
44/44 [==============================] - 30s 677ms/step - loss: 1.0335 - accuracy: 0.5898
Epoch 5/15
44/44 [==============================] - 28s 632ms/step - loss: 0.9947 - accuracy: 0.6141
Epoch 6/15
44/44 [==============================] - 29s 645ms/step - loss: 0.9548 - accuracy: 0.6291
Epoch 7/15
44/44 [==============================] - 28s 627ms/step - loss: 0.9056 - accuracy: 0.6530
Epoch 8/15
44/44 [==============================] - 29s 651ms/step - loss: 0.8761 - accuracy: 0.6595
Epoch 9/15
44/44 [==============================] - 28s 632ms/step - loss: 0.8754 - accuracy: 0.6606
Epoch 10/15
44/44 [==============================] - 28s 636ms/step - loss: 0.8156 - accuracy: 0.6894
Epoch 11/15
44/44 [==============================] - 28s 637ms/step - loss: 0.8073 - accuracy: 0.6829
Epoch 12/15
44/44 [==============================] - 28s 635ms/step - loss: 0.7893 - accuracy: 0.6979
Epoch 13/15
44/44 [==============================] - 31s 691ms/step - loss: 0.8023 - accuracy: 0.6850
Epoch 14/15
44/44 [==============================] - 28s 634ms/step - loss: 0.7658 - accuracy: 0.7063
Epoch 15/15
44/44 [==============================] - 28s 632ms/step - loss: 0.7698 - accuracy: 0.7030
<keras.callbacks.History at 0x7f0865c15750>
```

Question-7:

Save The Model

Solution:

model.save("flowers.h5")

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

Question-8:

Test The Model

Solution:

import numpy as np
from tensorflow.keras.preprocessing import image
Rose = image.load_img('/content/flowers/rose/1562198683_8cd8cb5876_n.jpg',target_size=(200,210))
Rose

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

```
Rose = image.load_img('/content/flowers/rose/1562198683_8cd8cb5876_n.jpg',target_size=(200,210))
```

```
Rose
```

array = image.img_to_array(Rose)
array

```
array = image.img_to_array(Rose)
array
```

```
array([[[ 48.,  68.,  31.],
        [ 48.,  68.,  31.],
        [ 48.,  68.,  31.],
        ...,
        [ 19.,  34.,  15.],
        [ 20.,  35.,  16.],
        [ 21.,  36.,  17.]],

       [[ 48.,  68.,  31.],
        [ 48.,  68.,  31.],
        [ 48.,  68.,  31.],
        ...,
        [ 21.,  34.,  16.],
        [ 21.,  34.,  16.],
        [ 21.,  34.,  16.]],

       [[ 48.,  68.,  31.],
        [ 48.,  68.,  31.],
        [ 48.,  68.,  31.],
        ...,
        [ 21.,  34.,  16.],
        [ 21.,  34.,  16.],
        [ 21.,  34.,  16.]],

       ...,

       [[ 42.,  51.,  22.],
        [ 43.,  52.,  23.],
        [ 43.,  52.,  23.],
        ...,
        [191.,  27.,  62.],
        [188.,  24.,  59.],
        [183.,  19.,  46.]],

       [[ 42.,  51.,  22.],
        [ 43.,  52.,  23.],
        [ 43.,  52.,  23.],
        ...,
        [187.,  27.,  61.],
        [187.,  26.,  60.],
        [185.,  22.,  49.]],

       [[ 42.,  51.,  22.],
        [ 43.,  52.,  23.],
        [ 43.,  52.,  23.],
        ...,
        [183.,  25.,  60.],
        [181.,  21.,  55.],
        [179.,  19.,  45.]]], dtype=float32)
```

array = np.expand_dims(array,axis=0)
array

```
array = np.expand_dims(array,axis=0)
array
```

```
array([[[[ 48.,  68.,  31.],
         [ 48.,  68.,  31.],
         [ 48.,  68.,  31.],
         ...,
         [ 19.,  34.,  15.],
         [ 20.,  35.,  16.],
         [ 21.,  36.,  17.]],

        [[ 48.,  68.,  31.],
         [ 48.,  68.,  31.],
         [ 48.,  68.,  31.],
         ...,
         [ 21.,  34.,  16.],
         [ 21.,  34.,  16.],
         [ 21.,  34.,  16.]],

        [[ 48.,  68.,  31.],
         [ 48.,  68.,  31.],
         [ 48.,  68.,  31.],
         ...,
         [ 21.,  34.,  16.],
         [ 21.,  34.,  16.],
         [ 21.,  34.,  16.]],

        ...,

        [[ 42.,  51.,  22.],
         [ 43.,  52.,  23.],
         [ 43.,  52.,  23.],
         ...,
         [191.,  27.,  62.],
         [188.,  24.,  59.],
         [183.,  19.,  46.]],

        [[ 42.,  51.,  22.],
         [ 43.,  52.,  23.],
         [ 43.,  52.,  23.],
         ...,
         [187.,  27.,  61.],
         [187.,  26.,  60.],
         [185.,  22.,  49.]],

        [[ 42.,  51.,  22.],
         [ 43.,  52.,  23.],
         [ 43.,  52.,  23.],
         ...,
         [183.,  25.,  60.],
         [181.,  21.,  55.],
         [179.,  19.,  45.]]]], dtype=float32)
```

x_train.class_indices

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

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

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