Assignment -3

BUILD AN CNN MODEL FOR THE CLASSIFICATION OF FLOWERS

Assignment Date	10 October 2022
Student Name	Mr.S.K.GANAPATHYRAM
Student Roll Number	913319104019
Maximum Marks	2 Marks

Question-1:

DOWNLOAD THE DATA SET

!unzip '/content/Flowers-Dataset.zip'

```
!unzip '/content/Flowers-Dataset.zip'
  inflating: flowers/dandelion/5996421299_b9bf488c1a_n.jpg
  inflating: flowers/dandelion/6012046444_fd80afb63a_n.jpg
  inflating: flowers/dandelion/6019234426_d25ea1230a_m.jpg
  inflating: flowers/dandelion/6035460327_4bbb708eab_n.jpg
  inflating: flowers/dandelion/6044710875_0459796d1b_m.jpg
  inflating: flowers/dandelion/6060576850_984176cf4f_n.jpg
  inflating: flowers/dandelion/6103898045_e066cdeedf_n.jpg
  inflating: flowers/dandelion/6104442744_ee2bcd32e7_n.jpg
  inflating: flowers/dandelion/61242541_a04395e6bc.jpg
  inflating: flowers/dandelion/6132275522_ce46b33c33_m.jpg
  inflating: flowers/dandelion/6146107825_45f708ecd7_n.jpg
  inflating: flowers/dandelion/6208857436_14a65fe4af_n.jpg
  inflating: flowers/dandelion/62293290_2c463891ff_m.jpg
  inflating: flowers/dandelion/6229634119_af5fec0a22.jpg
  inflating: flowers/dandelion/6250363717_17732e992e_n.jpg
  inflating: flowers/dandelion/6400843175_ef07053f8f_m.jpg
  inflating: flowers/dandelion/6412422565_ce61ca48a9_n.jpg
  inflating: flowers/dandelion/645330051_06b192b7e1.jpg
  inflating: flowers/dandelion/6495802659_98b57e0cca_m.jpg
  inflating: flowers/dandelion/674407101_57676c40fb.jpg
  inflating: flowers/dandelion/6888894675_524a6accab_n.jpg
  inflating: flowers/dandelion/6897671808_57230e04c5_n.jpg
  inflating: flowers/dandelion/6900157914_c3387c11d8.jpg
  inflating: flowers/dandelion/6901435398_b3192ff7f8_m.jpg
  inflating: flowers/dandelion/6918170172_3215766bf4_m.jpg
```

Question-2:

IMAGE AUGUMENTATION

Solution

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
xtrain = train_datagen.flow_from_directory('/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,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
xtrain = train_datagen.flow_from_directory('/content/flowers',
                                           target_size=(64,64),
                                           class_mode='categorical',
                                           batch_size=100)
xtrain
```

```
Found 4317 images belonging to 5 classes.
<keras.preprocessing.image.DirectoryIterator at 0x7f0656af6fd0>
```

Question-3:

CREATE MODEL

Solution

from tensorflow.keras.models import Sequential

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

Question-4:

ADD LAYERS (CONVOLUTION,MAX POOLING ,FLATTEN, DENSE(HIDDEN LAYERS) ,OUTPUT)

Solution

```
# Flatten layer
model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3))) # Convolution layer
model.add(MaxPooling2D(pool_size=(2,2))) # Max pooling layer
model.add(Flatten())
```

```
model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3))) # Convolution layer
model.add(MaxPooling2D(pool_size=(2,2))) # Max pooling layer
model.add(Flatten()) # Flatten layer
model
```

```
<keras.engine.sequential.Sequential at 0x7f0657db3710>
```

```
# Dense layers
model.add(Dense(300,activation='relu')) # Hidden layer
model.add(Dense(150,activation='relu')) # Hidden layer
model.add(Dense(5,activation='softmax')) # Output layer
```

```
[17] model.add(Dense(300,activation='relu')) # Hidden layer
     model.add(Dense(150,activation='relu')) # Hidden layer
     model.add(Dense(5,activation='softmax')) # Output layer

     <keras.engine.sequential.Sequential at 0x7f0657db3710>
```

Question-5:

COMPILE THE MODEL

Solution

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

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

Question-6:

FIT THE MODEL

Solution

```
model.fit(xtrain,
      steps_per_epoch=len(xtrain),
      epochs=10,
      )
```

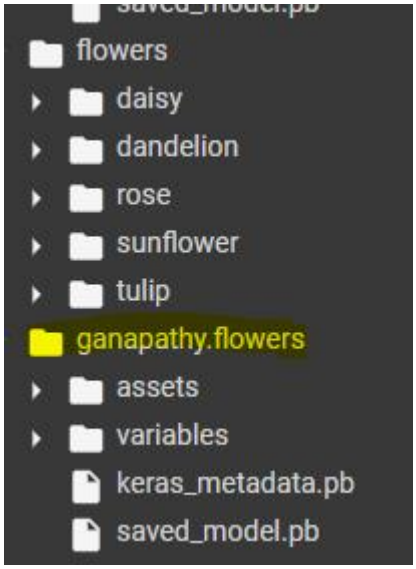
```
model.fit(xtrain,
          steps_per_epoch=len(xtrain),
          epochs=10,
          )

Epoch 1/10
44/44 [==============================] - 22s 292ms/step - loss: 1.6018 - accuracy: 0.2402
Epoch 2/10
44/44 [==============================] - 13s 303ms/step - loss: 1.6017 - accuracy: 0.2402
Epoch 3/10
44/44 [==============================] - 13s 301ms/step - loss: 1.5998 - accuracy: 0.2437
Epoch 4/10
44/44 [==============================] - 13s 300ms/step - loss: 1.5997 - accuracy: 0.2423
Epoch 5/10
44/44 [==============================] - 13s 304ms/step - loss: 1.6005 - accuracy: 0.2439
Epoch 6/10
44/44 [==============================] - 13s 301ms/step - loss: 1.5993 - accuracy: 0.2349
Epoch 7/10
44/44 [==============================] - 13s 302ms/step - loss: 1.6001 - accuracy: 0.2309
Epoch 8/10
44/44 [==============================] - 13s 303ms/step - loss: 1.5996 - accuracy: 0.2437
Epoch 9/10
44/44 [==============================] - 13s 298ms/step - loss: 1.5996 - accuracy: 0.2437
Epoch 10/10
44/44 [==============================] - 13s 301ms/step - loss: 1.5996 - accuracy: 0.2437
<keras.callbacks.History at 0x7f0640358550>
```

Question-7:

SAVE THE MODEL

Solution

```
model.save('ganapathy.flowers')
```

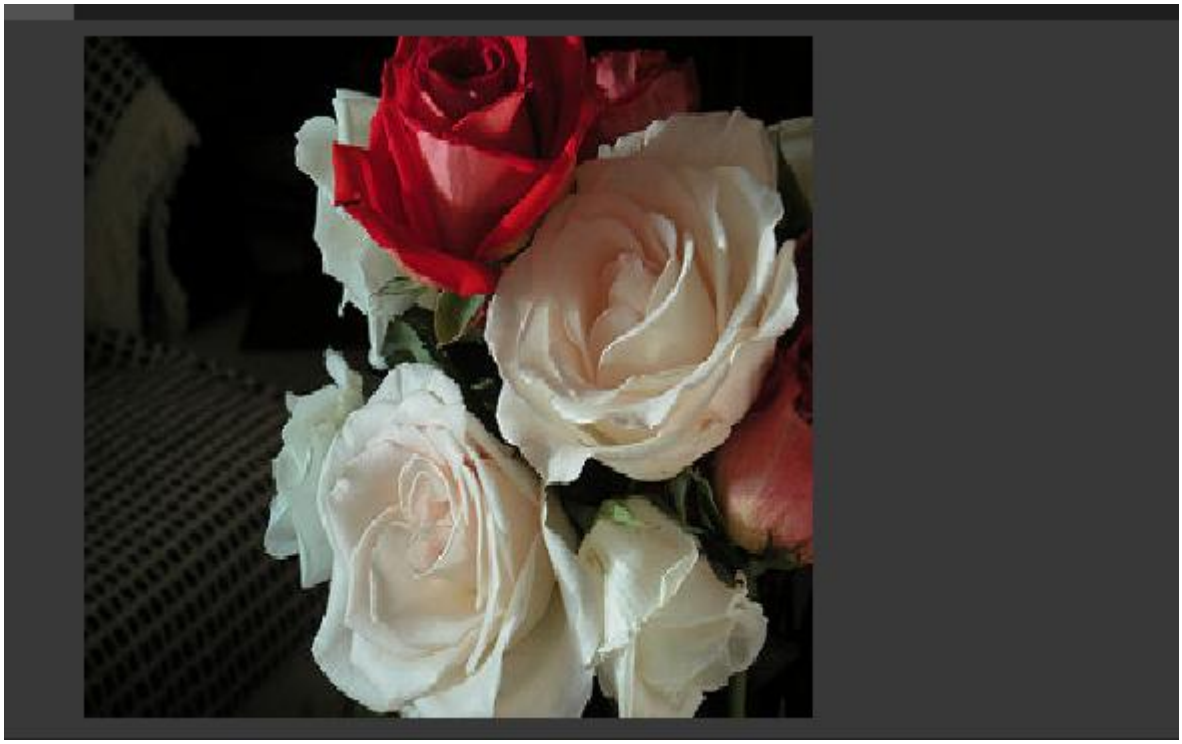


Question-8:

TEST THE MODEL

Solution

```
import numpy as np

from tensorflow.keras.preprocessing import image

img=image.load_img('/content/flowers/rose/102501987_3cdb8e5394_n.jpg',target_size=(340,364,3)
)

Img
```



```
x = image.img_to_array(img)

X
```

```
    ...,

    [[ 3.,  3.,  1.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     ...,
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.]],

    [[ 3.,  3.,  1.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     ...,
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.]],

    [[ 5.,  5.,  3.],
     [ 5.,  5.,  3.],
     [ 6.,  6.,  4.],
     ...,
     [ 2.,  2.,  0.],
     [ 3.,  3.,  1.],
     [ 3.,  3.,  1.]]], dtype=float32)
```

```
array([[[15., 16., 10.],
        [21., 22., 17.],
        [17., 17., 15.],
        ...,
        [ 2.,  2.,  0.],
        [ 2.,  2.,  0.],
        [ 1.,  1.,  0.]],

       [[20., 21., 15.],
        [13., 14.,  9.],
        [ 6.,  6.,  4.],
        ...,
        [ 2.,  2.,  0.],
        [ 2.,  2.,  0.],
        [ 2.,  2.,  0.]],

       [[20., 21., 15.],
        [13., 14.,  9.],
        [ 6.,  6.,  4.],
        ...,
        [ 2.,  2.,  0.],
        [ 2.,  2.,  0.],
        [ 2.,  2.,  0.]],

       ...,
```

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

X

```
array([[[[15., 16., 10.],
         [21., 22., 17.],
         [17., 17., 15.],
         ...,
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.],
         [ 1.,  1.,  0.]],

        [[20., 21., 15.],
         [13., 14.,  9.],
         [ 6.,  6.,  4.],
         ...,
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]],

        [[20., 21., 15.],
         [13., 14.,  9.],
         [ 6.,  6.,  4.],
         ...,
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.],
         [ 2.,  2.,  0.]],

        ...,
```

```
    [[ 3.,  3.,  1.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     ...,
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.]],

    [[ 3.,  3.,  1.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     ...,
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.],
     [ 2.,  2.,  0.]],

    [[ 5.,  5.,  3.],
     [ 5.,  5.,  3.],
     [ 6.,  6.,  4.],
     ...,
     [ 2.,  2.,  0.],
     [ 3.,  3.,  1.],
     [ 3.,  3.,  1.]]]], dtype=float32)
```

xtrain.class_indices

```
xtrain.class_indices
```

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

op = ['daisy','dandelion','rose','sunflower','tulip']
pred = np.argmax(model.predict(x))
op[pred]

```
'dandelion'
```

img=image.load_img('/content/flowers/rose/2325232198_751645d0bb_n.jpg',target_size=(300,300))

Img