Assignment – 3

Assignment Date	02 October 2022
Student Name	V.gokulakrishnan
Student Roll Number	820319205011
Maximum Marks	2 Marks

Pre-Requisites

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

Mounted at /content/gdrive

1. UNZIP FILES

```
cd/content/gdrive/MyDrive/CNN
```

OUTPUT: /content/gdrive/MyDrive/CNN

```
Pwd
```

OUTPUT: /content

```
!unzip Flowers-Dataset.zip
```

OUTPUT:

```
!unzip Flowers-Dataset.zip

Archive:  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
```

2. Image Augumentation

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

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

test_datagen=ImageDataGenerator(rescale=1./255)

x_train=train_datagen.flow_from_directory(r"/content/gdrive/MyDrive/CNN/flowe
rs",target_size=(64,64),class_mode='categorical',batch_size=24)
```

OUTPUT:

```
Found 4317 images belonging to 5 classes.

x_test=test_datagen.flow_from_directory(r"/content/gdrive/MyDrive/CNN/flowers
",target_size=(64,64),class_mode='categorical',batch_size=24)
```

OUTPUT:

```
Found 4317 images belonging to 5 classes.

x_train.class_indices
```

OUTPUT:

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

3. Initializing CNN And Create Model

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

4. Add layers

```
model=Sequential()
```

4.1 Input Layers (Convolution ,MaxPooling,Flatten)

```
model.add(Convolution2D(32,(3,3),activation='relu',strides=(1,1),input_shape=
(64,64,3)))

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

model.add(Flatten())

model.summary()

Model: "sequential"
```

OUTPUT:

```
_________________________________________________________________
```

```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)            (None, 30752)             0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

4.2 Hidden Layers

```
model.add(Dense(300,activation='relu'))
model.add(Dense(300,activation='relu'))
```

4.3 Output Layers

```
model.add(Dense(5,activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accu
racy'])

len(x_train)
```

OUTPUT:

```
180
```

5. Train the Model

```
model.fit_generator(x_train,steps_per_epoch=len(x_train),
validation_data=x_test, validation_steps=len(x_test), epochs= 30)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version.
Please use `Model.fit`, which supports generators.
  """Entry point for launching an IPython kernel.
```

OUTPUT:

```
Epoch 1/30
180/180 [==============================] - 66s 365ms/step - loss: 1.3510 -
accuracy: 0.4716 - val_loss: 1.0987 - val_accuracy: 0.5455
Epoch 2/30
180/180 [==============================] - 65s 362ms/step - loss: 1.0457 -
```

```
accuracy: 0.5923 - val_loss: 0.9647 - val_accuracy: 0.6305
Epoch 3/30
180/180 [==============================] - 65s 363ms/step - loss: 0.9304 -
accuracy: 0.6414 - val_loss: 0.8766 - val_accuracy: 0.6560
Epoch 4/30
180/180 [==============================] - 66s 368ms/step - loss: 0.8652 -
accuracy: 0.6688 - val_loss: 0.7802 - val_accuracy: 0.7060
Epoch 5/30
180/180 [==============================] - 66s 367ms/step - loss: 0.8047 -
accuracy: 0.6998 - val_loss: 0.7506 - val_accuracy: 0.7167
Epoch 6/30
180/180 [==============================] - 66s 364ms/step - loss: 0.7546 -
accuracy: 0.7139 - val_loss: 0.6637 - val_accuracy: 0.7501
Epoch 7/30
180/180 [==============================] - 65s 364ms/step - loss: 0.6957 -
accuracy: 0.7450 - val_loss: 0.6843 - val_accuracy: 0.7429
Epoch 8/30
180/180 [==============================] - 66s 364ms/step - loss: 0.6613 -
accuracy: 0.7552 - val_loss: 0.6681 - val_accuracy: 0.7538
Epoch 9/30
180/180 [==============================] - 66s 366ms/step - loss: 0.6241 -
accuracy: 0.7684 - val_loss: 0.5065 - val_accuracy: 0.8110
Epoch 10/30
180/180 [==============================] - 65s 361ms/step - loss: 0.5817 -
accuracy: 0.7841 - val_loss: 0.5033 - val_accuracy: 0.8070
Epoch 11/30
180/180 [==============================] - 65s 362ms/step - loss: 0.5395 -
accuracy: 0.8024 - val_loss: 0.4473 - val_accuracy: 0.8314
Epoch 12/30
180/180 [==============================] - 66s 366ms/step - loss: 0.5124 -
accuracy: 0.8061 - val_loss: 0.4321 - val_accuracy: 0.8492
Epoch 13/30
180/180 [==============================] - 65s 363ms/step - loss: 0.4951 -
accuracy: 0.8205 - val_loss: 0.3971 - val_accuracy: 0.8562
Epoch 14/30
180/180 [==============================] - 65s 364ms/step - loss: 0.4294 -
accuracy: 0.8406 - val_loss: 0.3292 - val_accuracy: 0.8860
Epoch 15/30
180/180 [==============================] - 66s 365ms/step - loss: 0.4454 -
accuracy: 0.8358 - val_loss: 0.2689 - val_accuracy: 0.9073
Epoch 16/30
180/180 [==============================] - 65s 363ms/step - loss: 0.3713 -
accuracy: 0.8670 - val_loss: 0.3174 - val_accuracy: 0.8870
Epoch 17/30
180/180 [==============================] - 65s 360ms/step - loss: 0.3577 -
accuracy: 0.8661 - val_loss: 0.2839 - val_accuracy: 0.8988
Epoch 18/30
180/180 [==============================] - 65s 360ms/step - loss: 0.3376 -
accuracy: 0.8809 - val_loss: 0.2085 - val_accuracy: 0.9305
Epoch 19/30
```

```
180/180 [==============================] - 65s 362ms/step - loss: 0.3433 -
accuracy: 0.8819 - val_loss: 0.1769 - val_accuracy: 0.9338
Epoch 20/30
180/180 [==============================] - 65s 363ms/step - loss: 0.2851 -
accuracy: 0.8981 - val_loss: 0.1628 - val_accuracy: 0.9414
Epoch 21/30
180/180 [==============================] - 65s 362ms/step - loss: 0.2579 -
accuracy: 0.9115 - val_loss: 0.1369 - val_accuracy: 0.9527
Epoch 22/30
180/180 [==============================] - 65s 363ms/step - loss: 0.2618 -
accuracy: 0.9064 - val_loss: 0.1864 - val_accuracy: 0.9351
Epoch 23/30
180/180 [==============================] - 65s 360ms/step - loss: 0.2288 -
accuracy: 0.9212 - val_loss: 0.1617 - val_accuracy: 0.9439
Epoch 24/30
180/180 [==============================] - 66s 365ms/step - loss: 0.2011 -
accuracy: 0.9300 - val_loss: 0.1725 - val_accuracy: 0.9377
Epoch 25/30
180/180 [==============================] - 65s 364ms/step - loss: 0.2088 -
accuracy: 0.9270 - val_loss: 0.1762 - val_accuracy: 0.9356
Epoch 26/30
180/180 [==============================] - 64s 358ms/step - loss: 0.2160 -
accuracy: 0.9280 - val_loss: 0.1351 - val_accuracy: 0.9551
Epoch 27/30
180/180 [==============================] - 65s 361ms/step - loss: 0.1996 -
accuracy: 0.9317 - val_loss: 0.1644 - val_accuracy: 0.9421
Epoch 28/30
180/180 [==============================] - 65s 361ms/step - loss: 0.1882 -
accuracy: 0.9363 - val_loss: 0.1495 - val_accuracy: 0.9479
Epoch 29/30
180/180 [==============================] - 65s 363ms/step - loss: 0.1568 -
accuracy: 0.9504 - val_loss: 0.1157 - val_accuracy: 0.9613
Epoch 30/30
180/180 [==============================] - 65s 361ms/step - loss: 0.1678 -
accuracy: 0.9467 - val_loss: 0.1115 - val_accuracy: 0.9620

<keras.callbacks.History at 0x7fac57b46510>
```

6. Save The model

```
model.save('Flowers_classification.h5')
```

7. Test The model

```
Ls
```

OUTPUT:

```
flowers/  Flowers_classification_model1.h5  Flowers-Dataset.zip  video.mp4
```

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

# Load the model
model=load_model('Flowers_classification.h5')


img=image.load_img(r"/content/gdrive/MyDrive/CNN/flowers/sunflower/1038652569
5_2c38fea555_n.jpg",target_size=(64,64))


img
```

OUTPUT:

```
x = image.img_to_array(img)

x
```

OUTPUT:

```
array([[[234.,  70.,  94.],
        [185., 124., 119.],
        [156.,  75.,  81.],
        ...,
        [ 82.,  67.,  86.],
        [111.,  24.,  33.],
        [220.,  96.,  98.]],

       [[220., 164., 173.],
        [203.,  68.,  82.],
        [178., 107.,  85.],
        ...,
        [ 29.,   5.,  29.],
        [192.,  91.,  83.],
        [144.,  80.,  78.]],

       [[167.,  78.,  60.],
        [236., 162., 161.],
        [155.,  69.,  52.],
```

```
        ...,
        [165.,  87.,  67.],
        [145.,  85.,  75.],
        [134.,  81.,  65.]],

       ...,

       [[ 33.,   0.,   0.],
        [118.,  46.,  34.],
        [201.,  55.,  32.],
        ...,
        [ 98.,  69.,  55.],
        [100.,  71.,  57.],
        [ 94.,  81.,  73.]],

       [[136.,  44.,  29.],
        [188.,  99.,  59.],
        [160.,  78.,  56.],
        ...,
        [ 14.,   1.,  11.],
        [  8.,   0.,  13.],
        [  6.,   0.,  18.]],

       [[118., 103.,  70.],
        [108.,  69.,  12.],
        [100.,  77.,  25.],
        ...,
        [ 41.,  40.,   9.],
        [ 46.,  42.,  13.],
        [ 41.,  41.,   5.]]], dtype=float32)
```

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

x
```

OUTPUT:

```
array([[[[234.,  70.,  94.],
         [185., 124., 119.],
         [156.,  75.,  81.],
         ...,
         [ 82.,  67.,  86.],
         [111.,  24.,  33.],
         [220.,  96.,  98.]],

        [[220., 164., 173.],
         [203.,  68.,  82.],
         [178., 107.,  85.],
```

```
        ...,
        [ 29.,   5.,  29.],
        [192.,  91.,  83.],
        [144.,  80.,  78.]],

       [[167.,  78.,  60.],
        [236., 162., 161.],
        [155.,  69.,  52.],
        ...,
        [165.,  87.,  67.],
        [145.,  85.,  75.],
        [134.,  81.,  65.]],

       ...,

       [[ 33.,   0.,   0.],
        [118.,  46.,  34.],
        [201.,  55.,  32.],
        ...,
        [ 98.,  69.,  55.],
        [100.,  71.,  57.],
        [ 94.,  81.,  73.]],

       [[136.,  44.,  29.],
        [188.,  99.,  59.],
        [160.,  78.,  56.],
        ...,
        [ 14.,   1.,  11.],
        [  8.,   0.,  13.],
        [  6.,   0.,  18.]],

       [[118., 103.,  70.],
        [108.,  69.,  12.],
        [100.,  77.,  25.],
        ...,
        [ 41.,  40.,   9.],
        [ 46.,  42.,  13.],
        [ 41.,  41.,   5.]]]], dtype=float32)
```

```
pred = model.predict(x)
```

```
pred
```

OUTPUT:

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

```
x_train.class_indices
```

OUTPUT:

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

```
index=['daisy','dandelion','rose','sunflower','tulip']
index[np.argmax(pred)]
```

OUTPUT:

‘daisy’