Assignment -3

Build CNN Model for Classification of Flowers

Assignment Date	:	06-10-2022
Student Name	:	S. Nagu
Student Roll Number	:	912419104020
Project	:	Fertilizer Recommendation System for Disease Prediction
Maximum Marks	:	2 Marks

Question-1:

Download the 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/Classroom/Flowers-Dataset.zip'
```

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

Archive:  /content/drive/MyDrive/Classroom/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
  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
```

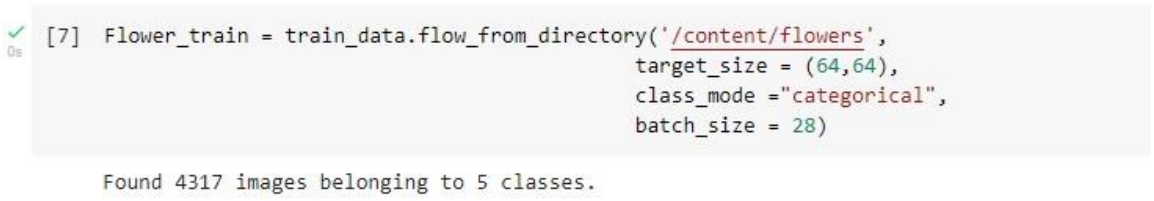
Question-2:

Image Augmentation

Solution:

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

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

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


```
Flower_train = train_data.flow_from_directory('/content/flowers',
                                              target_size = (64,64),
                                              class_mode ="categorical",
                                              batch_size = 28)
```

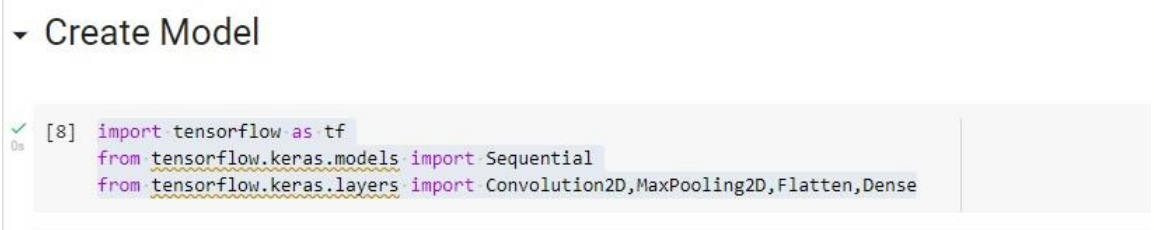


Question-3:

Create Model

Solution:

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



model=Sequential()

```
[9] model=Sequential()
```

Question-4:

Add Layers(Convolution,Maxpolling,Flatten,Dense-(Hidden Layers),Output)

Compile the model

Fit the model

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
```

model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
#fully connected layer
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
# output layer
model.add(Dense(5,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(Flower_train,steps_per_epoch=len(Flower_train),validation_data= Flower_test,
validation_steps=len(Flower_test),epochs=10)

In [49]:
```
model=Sequential()
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
#fully connected layer
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
# output layer
model.add(Dense(5,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics
=['accuracy'])
model.fit(Flower_train,steps_per_epoch=len(Flower_train),validation_data=Flower_test,
validation_steps=len(Flower_test),epochs=10)
```

```
Epoch 1/10
155/155 [==============================] - 53s 342ms/step - loss: 1.4066 - accuracy: 0.4281 - val_loss: 1.1024 - val_accuracy: 0.5613
Epoch 2/10
155/155 [==============================] - 49s 313ms/step - loss: 1.0564 - accuracy: 0.5777 - val_loss: 1.0354 - val_accuracy: 0.6025
Epoch 3/10
155/155 [==============================] - 48s 309ms/step - loss: 0.9837 - accuracy: 0.6125 - val_loss: 0.9683 - val_accuracy: 0.6310
Epoch 4/10
155/155 [==============================] - 48s 307ms/step - loss: 0.9243 - accuracy: 0.6407 - val_loss: 0.8618 - val_accuracy: 0.6692
Epoch 5/10
155/155 [==============================] - 48s 311ms/step - loss: 0.8944 - accuracy: 0.6472 - val_loss: 0.8043 - val_accuracy: 0.6933
Epoch 6/10
155/155 [==============================] - 49s 314ms/step - loss: 0.8485 - accuracy: 0.6769 - val_loss: 0.8472 - val_accuracy: 0.6750
Epoch 7/10
155/155 [==============================] - 49s 315ms/step - loss: 0.8464 - accuracy: 0.6676 - val_loss: 0.9060 - val_accuracy: 0.6669
Epoch 8/10
155/155 [==============================] - 51s 328ms/step - loss: 0.7957 - accuracy: 0.6952 - val_loss: 0.7423 - val_accuracy: 0.7109
Epoch 9/10
155/155 [==============================] - 48s 311ms/step - loss: 0.7748 - accuracy: 0.7035 - val_loss: 0.7088 - val_accuracy: 0.7299
Epoch 10/10
155/155 [==============================] - 50s 324ms/step - loss: 0.7530 - accuracy: 0.7104 - val_loss: 0.8266 - val_accuracy: 0.6915
```

Out[49]:
```
<keras.callbacks.History at 0x7f32ad7ffa50>
```

model.summary()

In [ ]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 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

 dense (Dense)               (None, 300)               9225900

 dense_1 (Dense)             (None, 150)               45150

 dense_2 (Dense)             (None, 5)                 755

=================================================================
Total params: 9,272,701
Trainable params: 9,272,701
Non-trainable params: 0
_________________________________________________________________
```

Question-7:

Save The Model

Solution:

model.save('Flower.h5')

▾ Save The Model

[ ]
```
model.save('Flower.h5')
```

Question-8:

Test The Model

Solution:

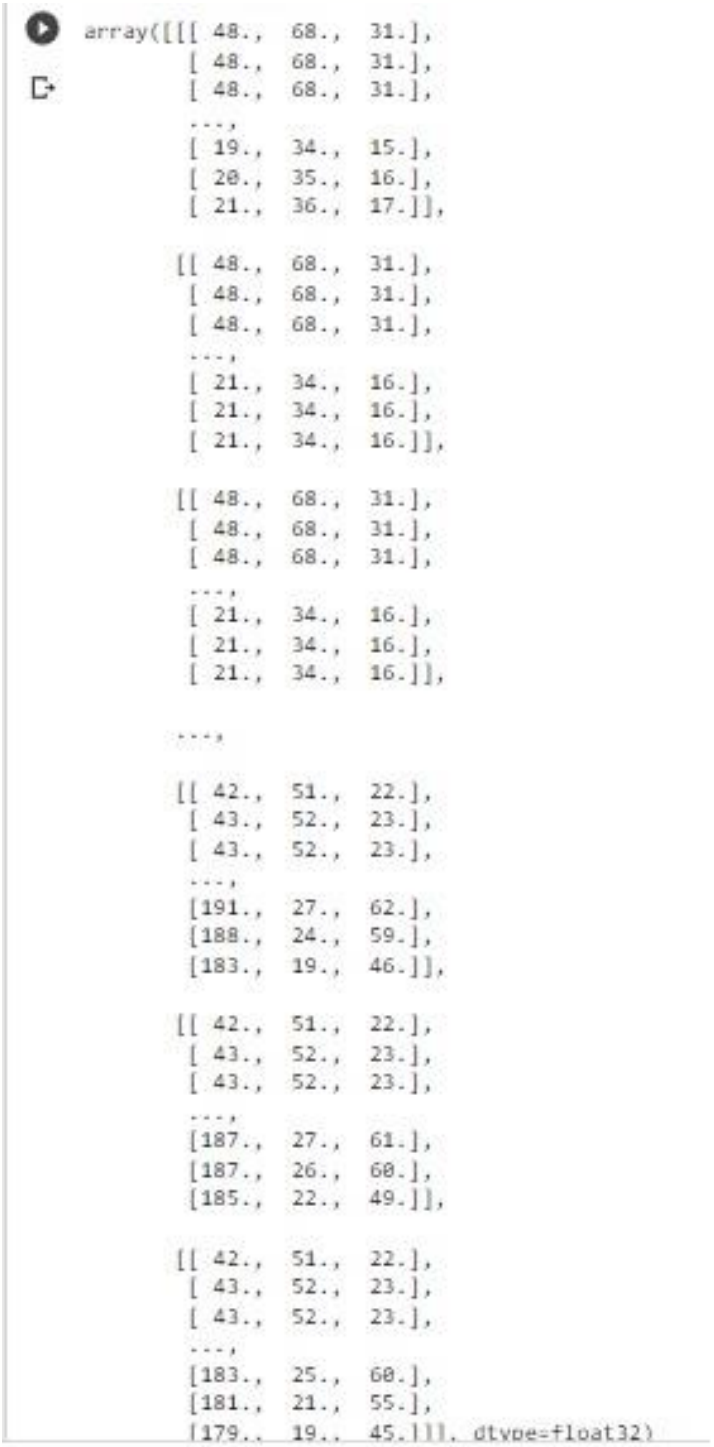
```
import numpy as np
from tensorflow.keras.preprocessing import image

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

rose
```

Test The Model

```
[ ] 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
```

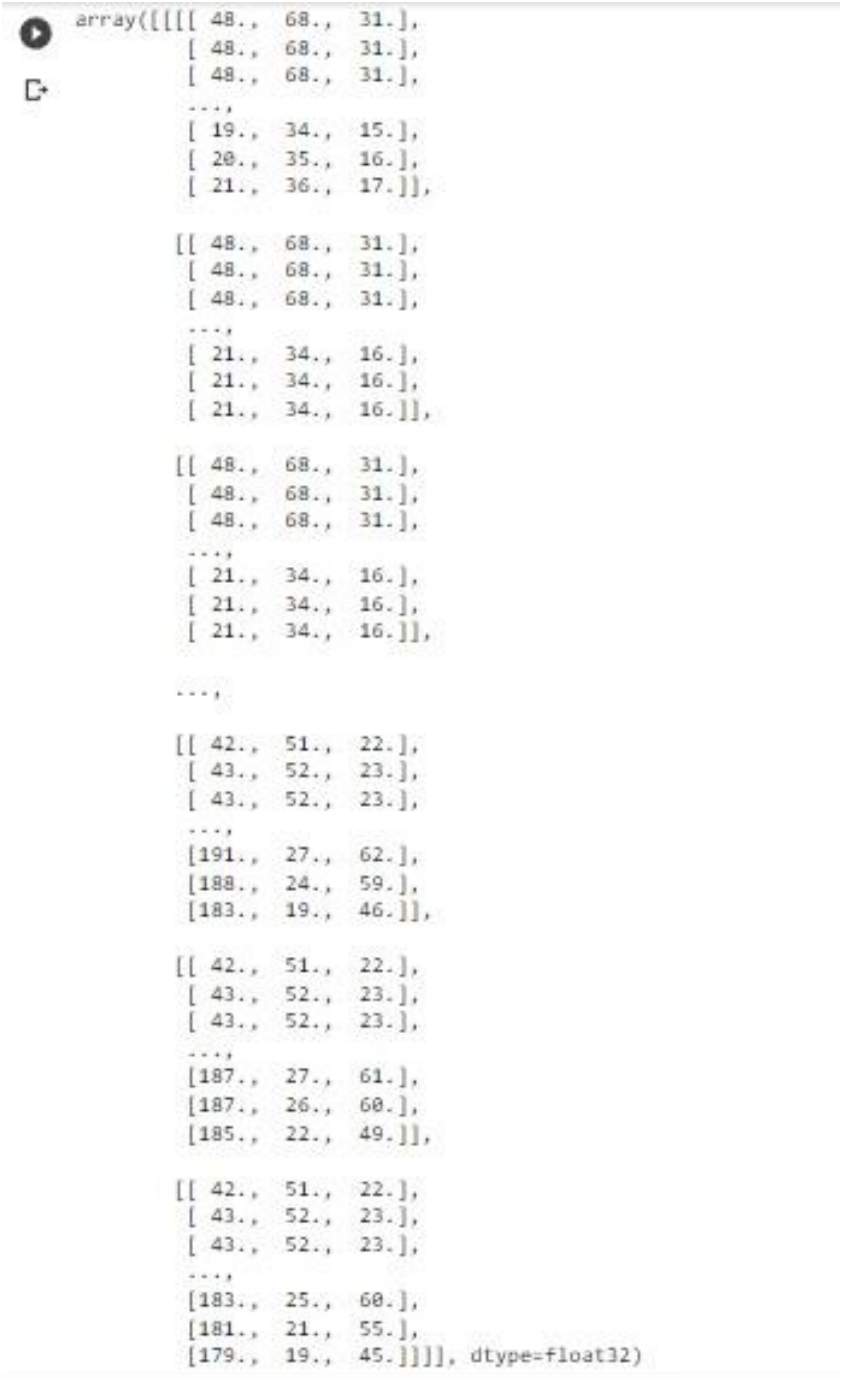
```
array = image.img_to_array(Rose)
array
```



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

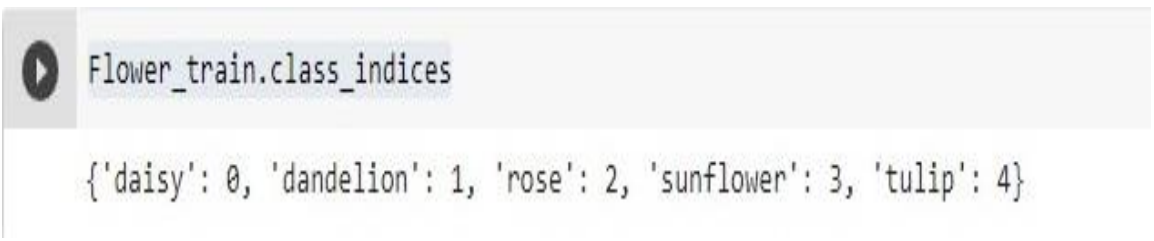
In [ ]:

```
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)
```

Flower_train.class_indices



```
Flower_train.class_indices
```

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

```
index=['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
index[2]
```

```
'rose'
```

```
dandelion = image.load_img('/content/flowers/dandelion/10043234166_e6dd915111_n.jpg',target_size=(64,64))
x = image.img_to_array(dandelion)
x = np.expand_dims(x,axis=0)
pred = np.argmax(model.predict(x))
op[pred]
```

```
'daisy'
```