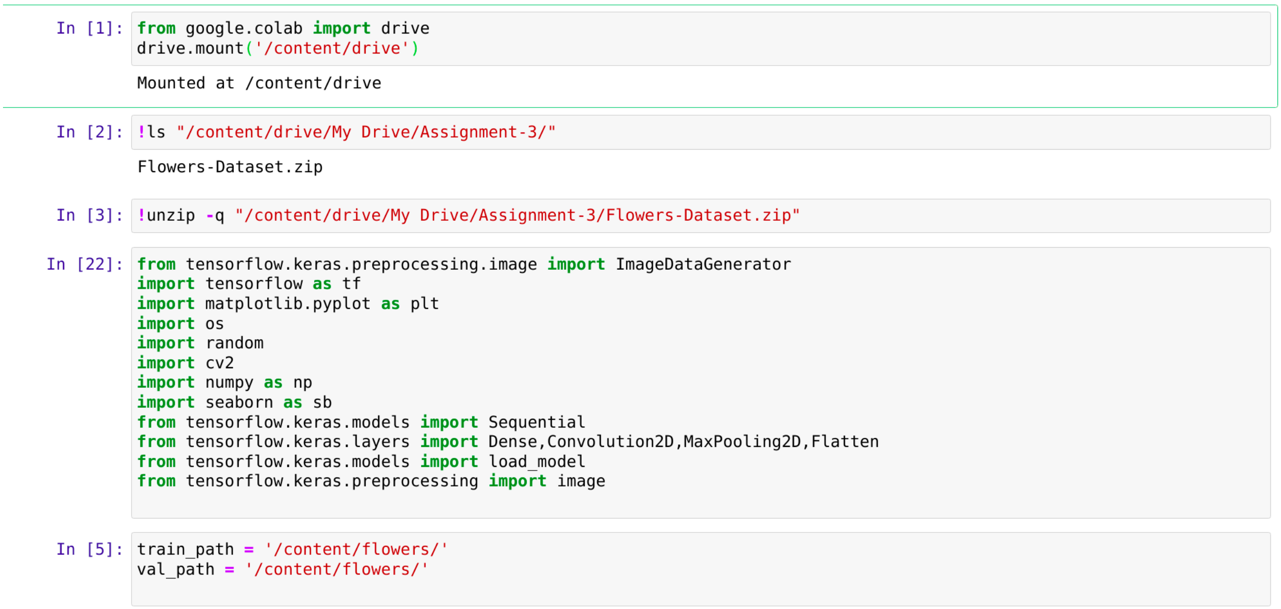
Assignment -3

Assignment Date	01 October 2022
Student Name	KALPIKA K
Student Roll Number	111519104058
Maximum Marks	2 Marks

Question:1

Download the Dataset



```
In [1]: from google.colab import drive
        drive.mount('/content/drive')

        Mounted at /content/drive

In [2]: !ls "/content/drive/My Drive/Assignment-3/"

        Flowers-Dataset.zip

In [3]: !unzip -q "/content/drive/My Drive/Assignment-3/Flowers-Dataset.zip"

In [22]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
         import tensorflow as tf
         import matplotlib.pyplot as plt
         import os
         import random
         import cv2
         import numpy as np
         import seaborn as sb
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense,Convolution2D,MaxPooling2D,Flatten
         from tensorflow.keras.models import load_model
         from tensorflow.keras.preprocessing import image

In [5]: train_path = '/content/flowers/'
        val_path = '/content/flowers/'
```

Question-2:

Image Augmentation

Solution:

data = ImageDataGenerator(rescale = 1.0/225, zoom_range = 0.2, horizontal_flip = True, vertical_flip = False, validation_split=0.25)
train_data = data.flow_from_directory('/content/flowers/', target_size=(224,224), class_mode = 'categorical', subset= 'training')
test_data = data.flow_from_directory('/content/flowers',target_size=(224,224),class_mode = 'categorical', subset = 'validation')
train_data.class_indices

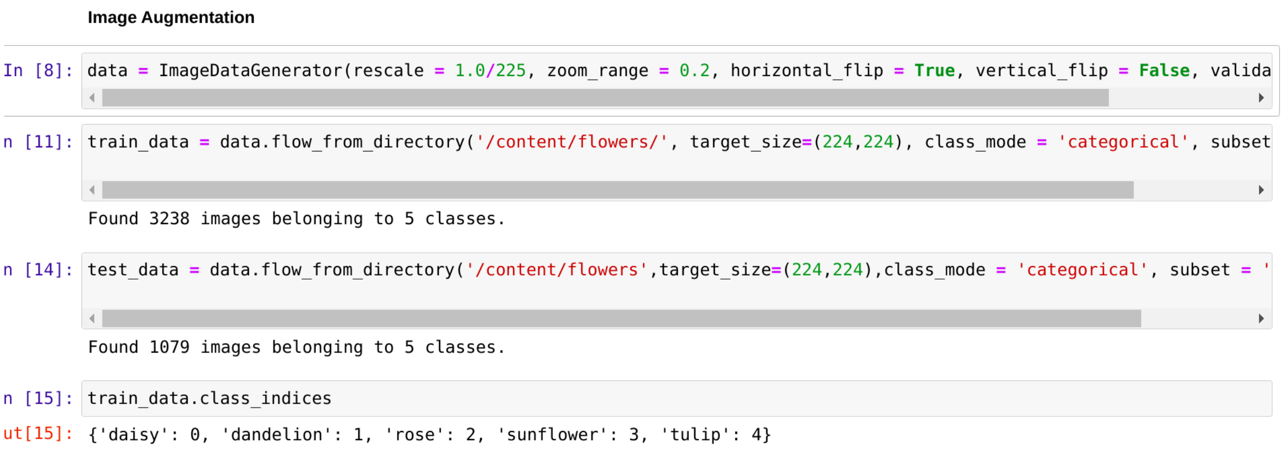


Image Augmentation

```
In [8]: data = ImageDataGenerator(rescale = 1.0/225, zoom_range = 0.2, horizontal_flip = True, vertical_flip = False, valida

n [11]: train_data = data.flow_from_directory('/content/flowers/', target_size=(224,224), class_mode = 'categorical', subset

        Found 3238 images belonging to 5 classes.

n [14]: test_data = data.flow_from_directory('/content/flowers',target_size=(224,224),class_mode = 'categorical', subset = '

        Found 1079 images belonging to 5 classes.

n [15]: train_data.class_indices

ut[15]: {'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

Question-3:

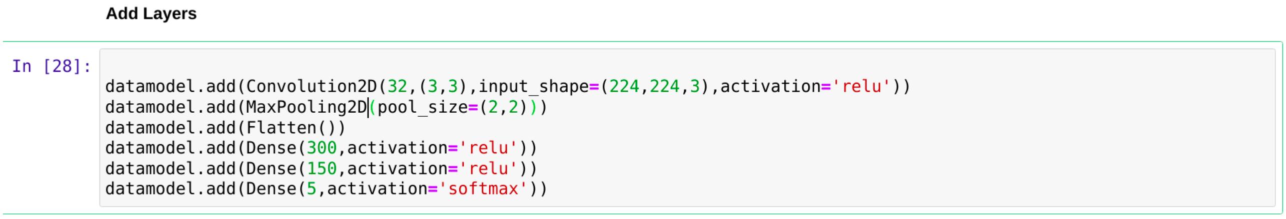
Create Model
Solution :
datamodel = Sequential()

Create Model

```
In [27]: datamodel = Sequential()
```

Question-4:

Add Layers
Solution:
datamodel.add(Convolution2D(32,(3,3),input_shape=(224,224,3),activation='relu'))
datamodel.add(MaxPooling2D(pool_size=(2,2)))
datamodel.add(Flatten())
datamodel.add(Dense(300,activation='relu'))
datamodel.add(Dense(150,activation='relu'))
datamodel.add(Dense(5,activation='softmax'))

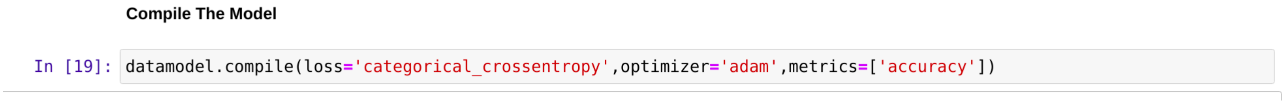


Add Layers

```
In [28]: 
         datamodel.add(Convolution2D(32,(3,3),input_shape=(224,224,3),activation='relu'))
         datamodel.add(MaxPooling2D(pool_size=(2,2)))
         datamodel.add(Flatten())
         datamodel.add(Dense(300,activation='relu'))
         datamodel.add(Dense(150,activation='relu'))
         datamodel.add(Dense(5,activation='softmax'))
```

Question-5:

Compile The Model
Soltion:
datamodel.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])



Compile The Model

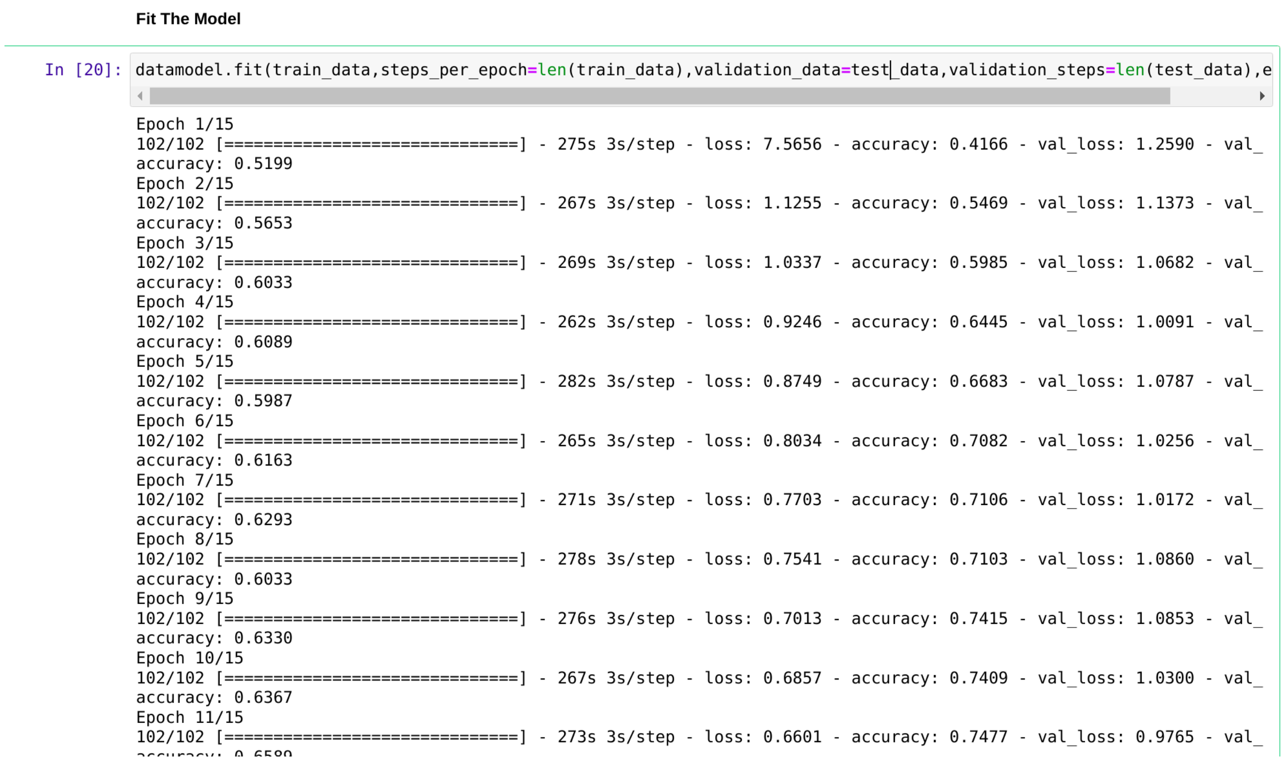
```
In [19]: datamodel.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Question -6 :

Fit The Model

Solution :

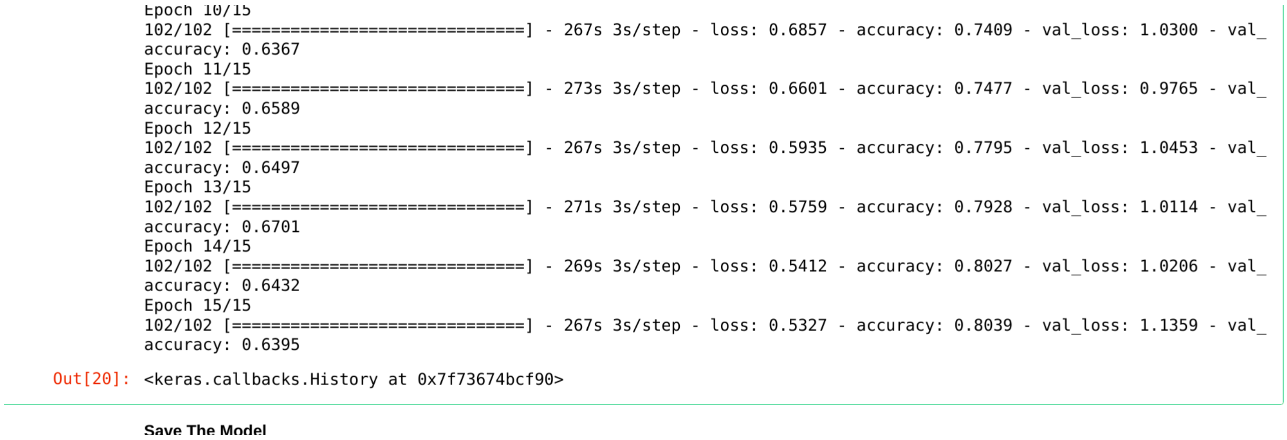
datamodel.fit(train_data,steps_per_epoch=len(train_data),validation_data=test_data,validation_steps=len(test_data),epochs=15)



Fit The Model

```
In [20]: datamodel.fit(train_data,steps_per_epoch=len(train_data),validation_data=test_data,validation_steps=len(test_data),e
```

```
Epoch 1/15
102/102 [==============================] - 275s 3s/step - loss: 7.5656 - accuracy: 0.4166 - val_loss: 1.2590 - val_
accuracy: 0.5199
Epoch 2/15
102/102 [==============================] - 267s 3s/step - loss: 1.1255 - accuracy: 0.5469 - val_loss: 1.1373 - val_
accuracy: 0.5653
Epoch 3/15
102/102 [==============================] - 269s 3s/step - loss: 1.0337 - accuracy: 0.5985 - val_loss: 1.0682 - val_
accuracy: 0.6033
Epoch 4/15
102/102 [==============================] - 262s 3s/step - loss: 0.9246 - accuracy: 0.6445 - val_loss: 1.0091 - val_
accuracy: 0.6089
Epoch 5/15
102/102 [==============================] - 282s 3s/step - loss: 0.8749 - accuracy: 0.6683 - val_loss: 1.0787 - val_
accuracy: 0.5987
Epoch 6/15
102/102 [==============================] - 265s 3s/step - loss: 0.8034 - accuracy: 0.7082 - val_loss: 1.0256 - val_
accuracy: 0.6163
Epoch 7/15
102/102 [==============================] - 271s 3s/step - loss: 0.7703 - accuracy: 0.7106 - val_loss: 1.0172 - val_
accuracy: 0.6293
Epoch 8/15
102/102 [==============================] - 278s 3s/step - loss: 0.7541 - accuracy: 0.7103 - val_loss: 1.0860 - val_
accuracy: 0.6033
Epoch 9/15
102/102 [==============================] - 276s 3s/step - loss: 0.7013 - accuracy: 0.7415 - val_loss: 1.0853 - val_
accuracy: 0.6330
Epoch 10/15
102/102 [==============================] - 267s 3s/step - loss: 0.6857 - accuracy: 0.7409 - val_loss: 1.0300 - val_
accuracy: 0.6367
Epoch 11/15
102/102 [==============================] - 273s 3s/step - loss: 0.6601 - accuracy: 0.7477 - val_loss: 0.9765 - val_
accuracy: 0.6589
```

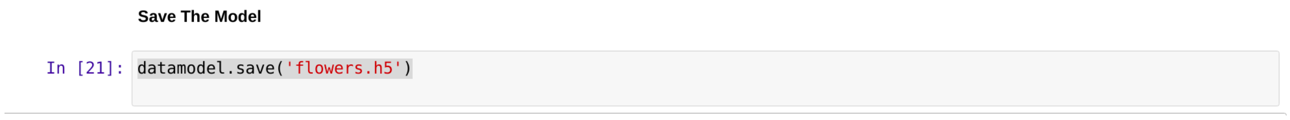


```
Epoch 10/15
102/102 [==============================] - 267s 3s/step - loss: 0.6857 - accuracy: 0.7409 - val_loss: 1.0300 - val_
accuracy: 0.6367
Epoch 11/15
102/102 [==============================] - 273s 3s/step - loss: 0.6601 - accuracy: 0.7477 - val_loss: 0.9765 - val_
accuracy: 0.6589
```

```
Epoch 12/15
102/102 [==============================] - 267s 3s/step - loss: 0.5935 - accuracy: 0.7795 - val_loss: 1.0453 - val_
accuracy: 0.6497
Epoch 13/15
102/102 [==============================] - 271s 3s/step - loss: 0.5759 - accuracy: 0.7928 - val_loss: 1.0114 - val_
accuracy: 0.6701
Epoch 14/15
102/102 [==============================] - 269s 3s/step - loss: 0.5412 - accuracy: 0.8027 - val_loss: 1.0206 - val_
accuracy: 0.6432
Epoch 15/15
102/102 [==============================] - 267s 3s/step - loss: 0.5327 - accuracy: 0.8039 - val_loss: 1.1359 - val_
accuracy: 0.6395
Out[20]: <keras.callbacks.History at 0x7f73674bcf90>
```

Save The Model

Question-7:
Save the model
solution :
datamodel.save('flowers.h5')



Save The Model

```
In [21]: datamodel.save('flowers.h5')
```

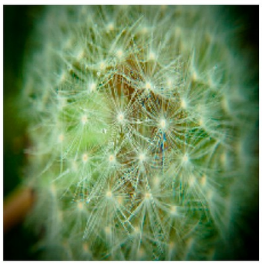
Question-8:

Test The Model
Solution:
img = image.load_img('/content/flowers/dandelion/10437652486_aa86c14985.jpg',target_size=(224,224))
temp_arr = image.img_to_array(img)
dim = np.expand_dims(temp_arr,axis=0)
temp=np.argmax(datamodel.predict(dim),axis=1)
index = ['Daisy','Dandelion','Rose','Sunflower','Tulip']
index[temp[0]]

Test The Model

```
In [23]: img = image.load_img('/content/flowers/dandelion/10437652486_aa86c14985.jpg',target_size=(224,224))
         img
```

Out[23]:



```
In [24]: temp_arr = image.img_to_array(img)
```

```
In [25]: dim = np.expand_dims(temp_arr,axis=0)
         temp=np.argmax(datamodel.predict(dim),axis=1)
         index = ['Daisy','Dandelion','Rose','Sunflower','Tulip']
         index[temp[0]]
```

```
Out[25]: 'Daisy'
```