Assignment -3

Build CNN Model for Classification Of Flowers

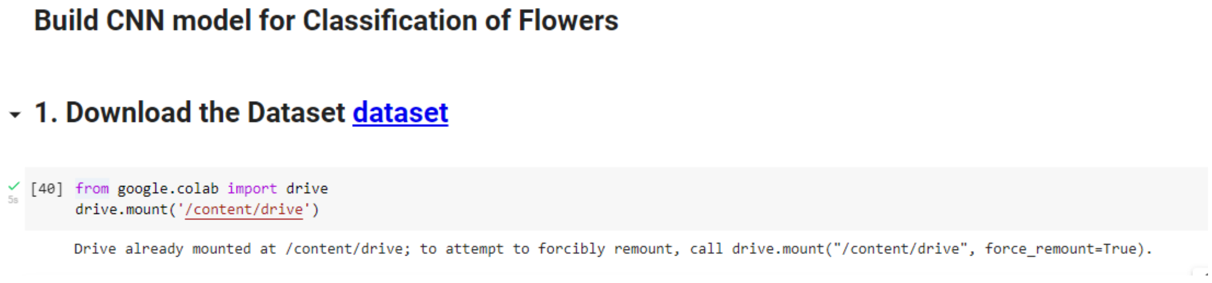
Assignment Date	09 October 2022
Student Name	Mythiresh G
Student Roll Number	310619205062
Maximum Marks	2 Marks

Task 1:

1. Download the Dataset : Dataset

Solution:

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

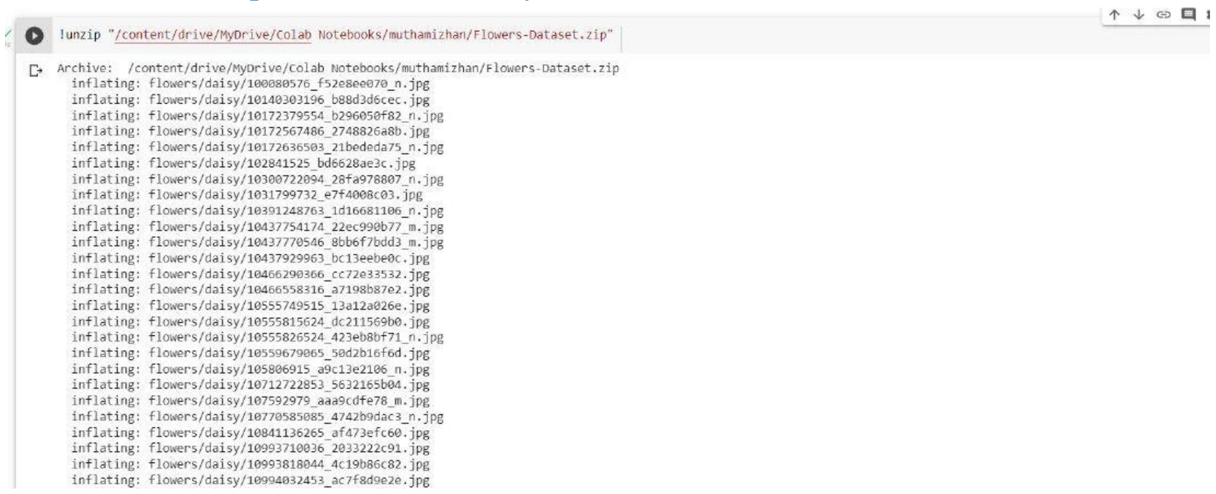


Build CNN model for Classification of Flowers

1. Download the Dataset dataset

```
[40] 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 the file
!unzip "/content/drive/MyDrive/Colab Notebooks/muthamizhan/Flowers-Dataset.zip"
```



```
!unzip "/content/drive/MyDrive/Colab Notebooks/muthamizhan/Flowers-Dataset.zip"

Archive:  /content/drive/MyDrive/Colab Notebooks/muthamizhan/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
```

Task 2:

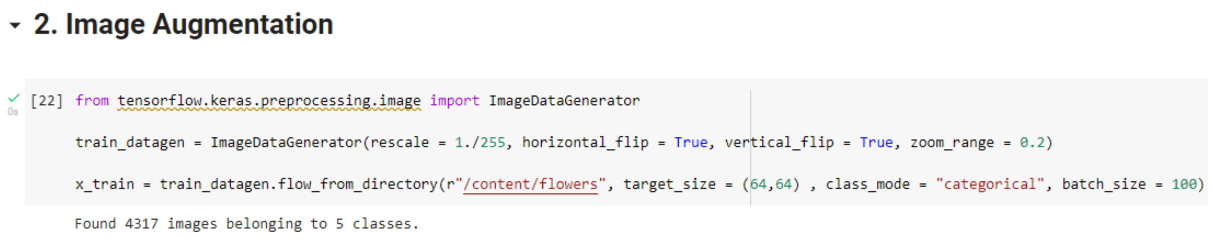
2. Image Augmentation

Solution:

from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

x_train = train_datagen.flow_from_directory(r"/content/flowers", target_size = (64,64) , class_mode = "categorical", batch_size = 100)
"



▾ 2. Image Augmentation

```
[22] from tensorflow.keras.preprocessing.image import ImageDataGenerator

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

     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.
```

Task 3:

3. Create Model

Solution:

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

model = Sequential()

▾ 3. Create Model

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

     model = Sequential()
```

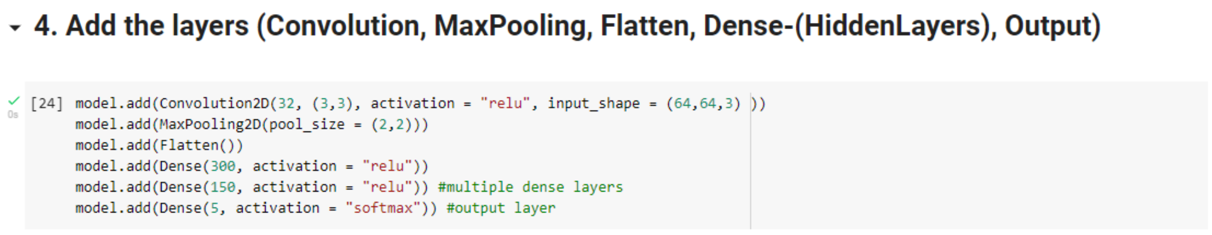
Task 4:

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

Solution:

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")) #multiple dense layers
model.add(Dense(5, activation = "softmax")) #output layer


▾ **4. Add the layers (Convolution, MaxPooling, Flatten, Dense-(HiddenLayers), Output)**

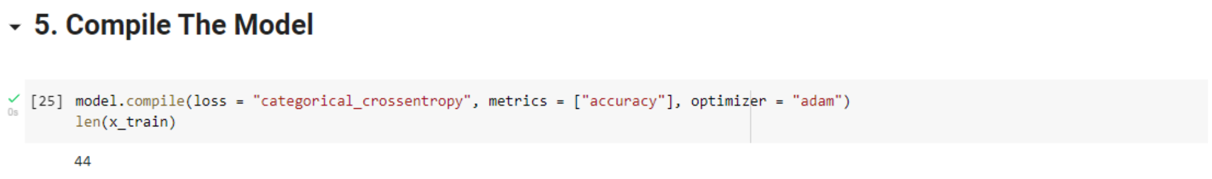
```
[24] model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3) ))
     model.add(MaxPooling2D(pool_size = (2,2)))
     model.add(Flatten())
     model.add(Dense(300, activation = "relu"))
     model.add(Dense(150, activation = "relu")) #multiple dense layers
     model.add(Dense(5, activation = "softmax")) #output layer
```

Task 5:

5. Compile The Model

Solution:

model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")
len(x_train)


▾ **5. Compile The Model**

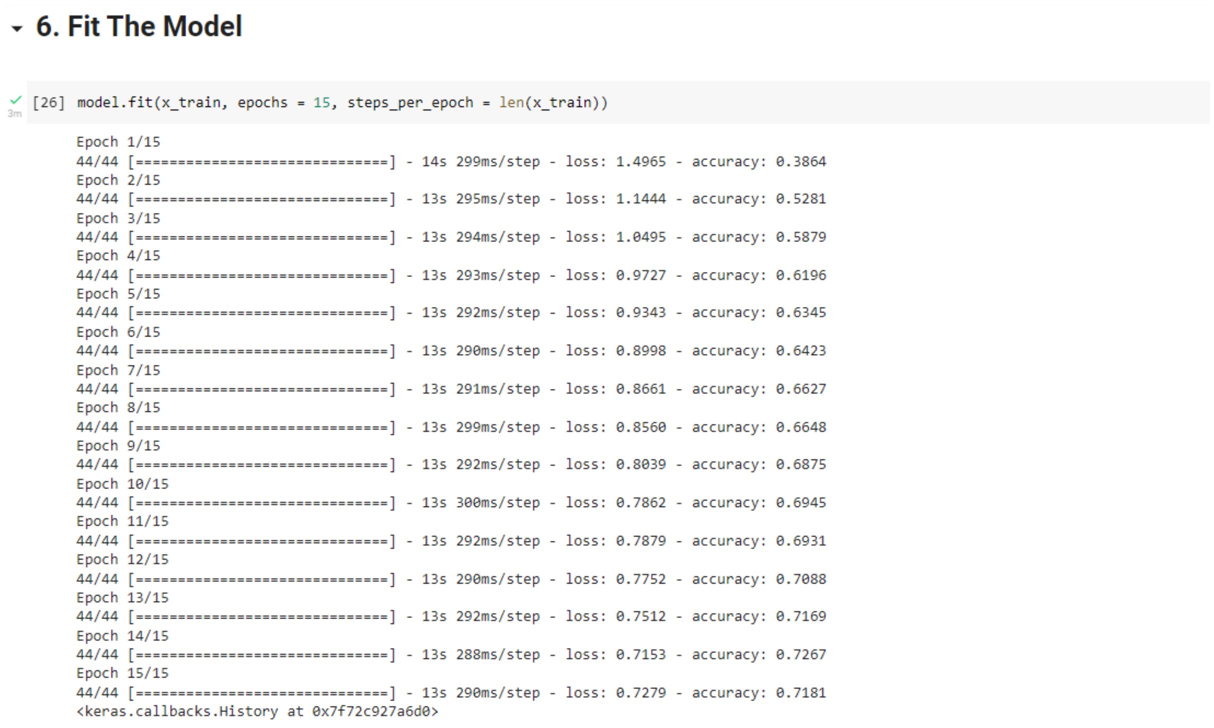
```
[25] model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")
     len(x_train)

     44
```

Task 6:

6. Fit The Model

Solution:

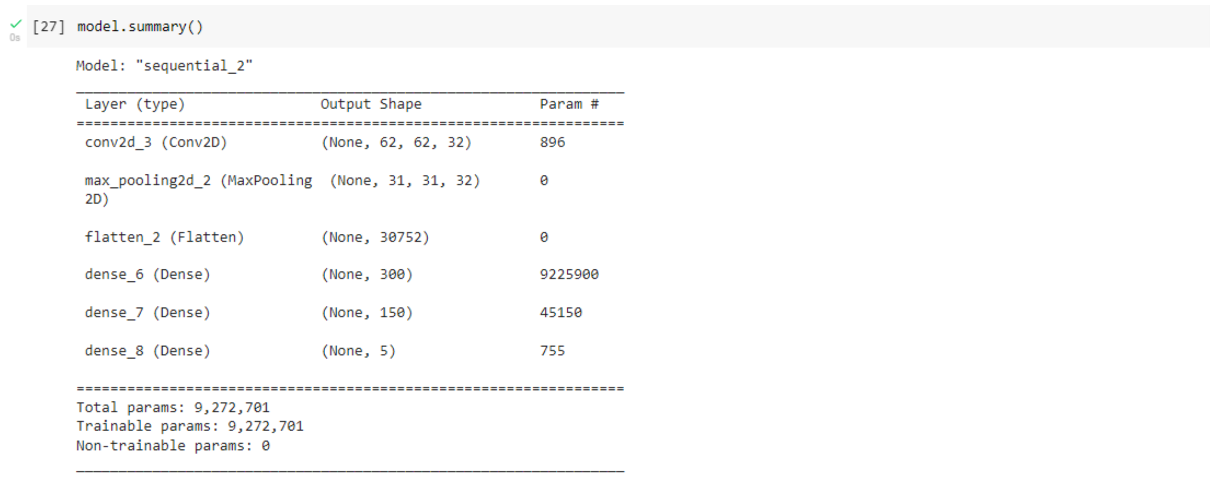
model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))



6. Fit The Model

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

Epoch 1/15
44/44 [==============================] - 14s 299ms/step - loss: 1.4965 - accuracy: 0.3864
Epoch 2/15
44/44 [==============================] - 13s 295ms/step - loss: 1.1444 - accuracy: 0.5281
Epoch 3/15
44/44 [==============================] - 13s 294ms/step - loss: 1.0495 - accuracy: 0.5879
Epoch 4/15
44/44 [==============================] - 13s 293ms/step - loss: 0.9727 - accuracy: 0.6196
Epoch 5/15
44/44 [==============================] - 13s 292ms/step - loss: 0.9343 - accuracy: 0.6345
Epoch 6/15
44/44 [==============================] - 13s 290ms/step - loss: 0.8998 - accuracy: 0.6423
Epoch 7/15
44/44 [==============================] - 13s 291ms/step - loss: 0.8661 - accuracy: 0.6627
Epoch 8/15
44/44 [==============================] - 13s 299ms/step - loss: 0.8560 - accuracy: 0.6648
Epoch 9/15
44/44 [==============================] - 13s 292ms/step - loss: 0.8039 - accuracy: 0.6875
Epoch 10/15
44/44 [==============================] - 13s 300ms/step - loss: 0.7862 - accuracy: 0.6945
Epoch 11/15
44/44 [==============================] - 13s 292ms/step - loss: 0.7879 - accuracy: 0.6931
Epoch 12/15
44/44 [==============================] - 13s 290ms/step - loss: 0.7752 - accuracy: 0.7088
Epoch 13/15
44/44 [==============================] - 13s 292ms/step - loss: 0.7512 - accuracy: 0.7169
Epoch 14/15
44/44 [==============================] - 13s 288ms/step - loss: 0.7153 - accuracy: 0.7267
Epoch 15/15
44/44 [==============================] - 13s 290ms/step - loss: 0.7279 - accuracy: 0.7181
<keras.callbacks.History at 0x7f72c927a6d0>
```

model.summary()



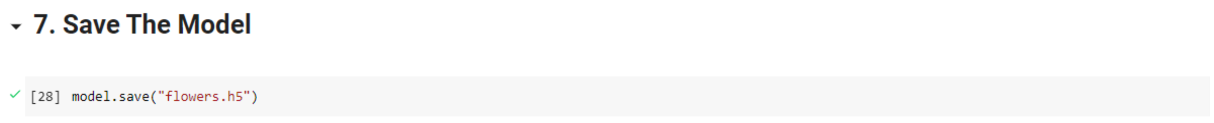
```
[27] model.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 62, 62, 32)        896

 max_pooling2d_2 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 flatten_2 (Flatten)         (None, 30752)             0

 dense_6 (Dense)             (None, 300)               9225900

 dense_7 (Dense)             (None, 150)               45150

 dense_8 (Dense)             (None, 5)                 755

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

Task 7:

7. Save The Model

Solution:

model.save("flowers.h5")



7. Save The Model

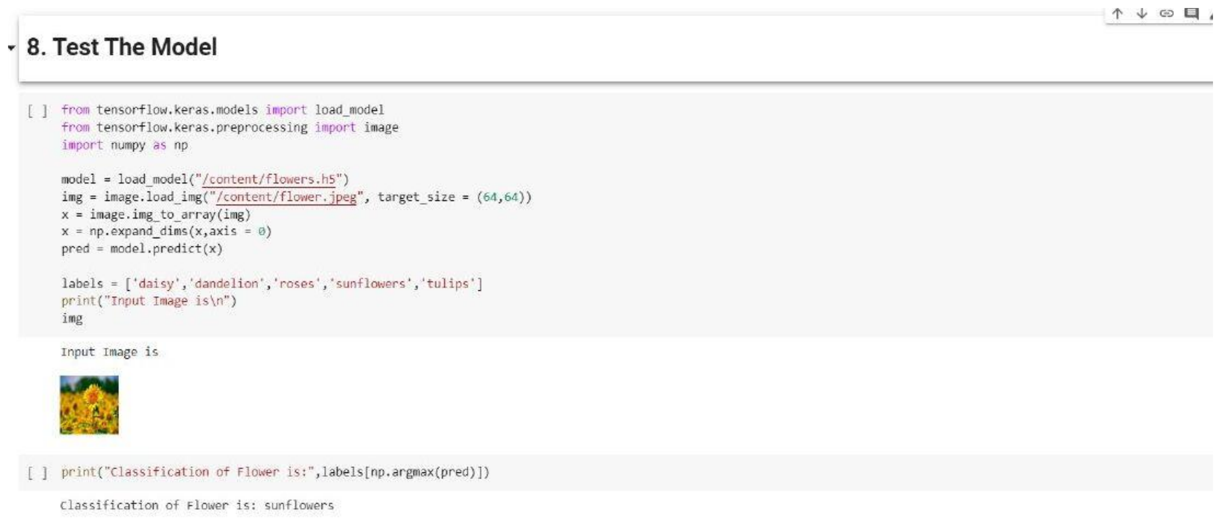
```
[28] model.save("flowers.h5")
```

Task 8:

9. Test The Model

Solution:

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

model = load_model("/content/flowers.h5")
img = image.load_img("/content/flower.jpe", target_size = (64,64))
x = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
pred = model.predict(x)

labels = ['daisy','dandelion','roses','sunflowers','tulips']
print("Input image is")
img

print("Classification of Flower is:",labels[np.argmax(pred)])
```



8. Test The Model

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

    model = load_model("/content/flowers.h5")
    img = image.load_img("/content/flower.jpeg", target_size = (64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)
    pred = model.predict(x)

    labels = ['daisy','dandelion','roses','sunflowers','tulips']
    print("Input Image is\n")
    img

    Input Image is
```

```
[ ] print("Classification of Flower is:",labels[np.argmax(pred)])

    Classification of Flower is: sunflowers
```