##**ASSIGNMENT_3** :- (Nithish.R.L)

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

```
Mounted at /content/drive
```

```
#Extracting Data
!unzip "/content/drive/MyDrive/Colab Notebooks/Flowers-Dataset
( Splitted ).zip"
```

##Image Augmentation :

```
#Import req. Lib.
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
#Augmentation On Training Variable
train_datagen = ImageDataGenerator(rescale= 1./255,
                 zoom_range=0.2,
                 horizontal_flip =True)
```

```
#Augmentation On Training Variable
test_datagen = ImageDataGenerator(rescale= 1./255)
```

```
#Augmentation On Training Variable
ftrain = train_datagen.flow_from_directory('/content/Flowers-Dataset (
Splitted )/Training',
                                           target_size=(64,64),
                                           class_mode='categorical',
                                           batch_size=100)
```

```
Found 4086 images belonging to 5 classes.
```

```
#Augmentation On Training Variable
ftest = test_datagen.flow_from_directory('/content/Flowers-Dataset
( Splitted )/Testing',
                                          target_size=(64,64),
                                          class_mode='categorical',
                                          batch_size=100)
```

```
Found 231 images belonging to 5 classes.
```

##Creating The Model :

Adding Layers :

```
#Import req. Lib.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D,
Flatten, Dense
```

```
# Build a CNN Block:
model = Sequential() #intializing sequential model
```

```
model.add(Convolution2D(32,(3,3),activation='relu',
input_shape=(64,64,3))) #convolution layer
model.add(MaxPooling2D(pool_size=(2, 2))) #Maxpooling layer
model.add(Flatten()) #Flatten layer
model.add(Dense(400,activation='relu')) #Hidden Layer 1
model.add(Dense(200,activation='relu')) #Hidden Layer 2
model.add(Dense(5,activation='softmax')) #Output Layer
```

Compiling :

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

Fit / Train The Model :

```
#Train Model:
model.fit_generator(ftrain,
                    steps_per_epoch=len(ftrain),
                    epochs=10,
                    validation_data=ftest,
                    validation_steps=len(ftest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.


Epoch 1/10
41/41 [==============================] - 30s 711ms/step - loss: 1.3702
- accuracy: 0.4315 - val_loss: 1.2850 - val_accuracy: 0.5238
Epoch 2/10
41/41 [==============================] - 29s 701ms/step - loss: 1.0398
- accuracy: 0.5918 - val_loss: 1.3570 - val_accuracy: 0.5411
Epoch 3/10
41/41 [==============================] - 29s 701ms/step - loss: 0.9675
- accuracy: 0.6238 - val_loss: 1.4026 - val_accuracy: 0.5065
Epoch 4/10
41/41 [==============================] - 29s 701ms/step - loss: 0.8853
- accuracy: 0.6601 - val_loss: 1.2253 - val_accuracy: 0.5887
Epoch 5/10
41/41 [==============================] - 29s 703ms/step - loss: 0.8395
- accuracy: 0.6806 - val_loss: 1.1541 - val_accuracy: 0.5801
Epoch 6/10
41/41 [==============================] - 29s 701ms/step - loss: 0.7740
- accuracy: 0.6982 - val_loss: 1.2437 - val_accuracy: 0.5714
Epoch 7/10
41/41 [==============================] - 29s 701ms/step - loss: 0.7467
- accuracy: 0.7181 - val_loss: 1.1862 - val_accuracy: 0.6277
Epoch 8/10
```

```
41/41 [==============================] - 29s 697ms/step - loss: 0.6988
- accuracy: 0.7332 - val_loss: 1.1816 - val_accuracy: 0.6061
Epoch 9/10
41/41 [==============================] - 29s 700ms/step - loss: 0.6728
- accuracy: 0.7442 - val_loss: 1.2922 - val_accuracy: 0.6104
Epoch 10/10
41/41 [==============================] - 30s 721ms/step - loss: 0.6166
- accuracy: 0.7624 - val_loss: 1.3966 - val_accuracy: 0.5931
```

```
<keras.callbacks.History at 0x7fd7e3c5eb90>
```

Saving The Model :

```
#Save Model
model.save('flowers.h5')
```

##Testing The Model :

```
#Import req. Lib.
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
#Testing No 1 :-
img = image.load_img('/content/Flowers-Dataset ( Splitted
)/Testing/daisy/34275662120_7757a15d07_n.jpg',target_size=(64,64))
#Reading image
f = image.img_to_array(img) #Convertinng image to array
f = np.expand_dims(f,axis=0) #Expanding dimensions
pred = np.argmax(model.predict(f)) #predicting higher propability
index
op = ['daisy','dandelion','rose','sunflower','tulip'] #Creating List
op[pred] #List indexing with output
```

```
{"type":"string"}
```

```
#Testing No 2 :-
img = image.load_img('/content/Flowers-Dataset ( Splitted
)/Testing/sunflower/14121915990_4b76718077_m.jpg',target_size=(64,64))
#Reading image
f = image.img_to_array(img) #Convertinng image to array
f = np.expand_dims(f,axis=0) #Expanding dimensions
pred = np.argmax(model.predict(f)) #predicting higher propability
index
op = ['daisy','dandelion','rose','sunflower','tulip'] #Creating List
op[pred] #List indexing with output
```

```
{"type":"string"}
```

```
#Testing No 3 :-
img = image.load_img('/content/Flowers-Dataset ( Splitted
)/Testing/tulip/19425920580_cdc8f49aed_n.jpg',target_size=(64,64))
#Reading image
```

```
f = image.img_to_array(img) #Convertinng image to array
f = np.expand_dims(f,axis=0) #Expanding dimensions
pred = np.argmax(model.predict(f)) #predicting higher propability index
op = ['daisy','dandelion','rose','sunflower','tulip'] #Creating List
op[pred] #List indexing with output
```

```
{"type":"string"}
```

For the above three tests performed the Model has predicted the images correctly..!