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

    Mounted at /content/drive

```
ls
```

    drive/ sample_data/

```
cd /content/drive/MyDrive/IBM_NalaiyaThiran/images
```

    /content/drive/MyDrive/IBM_NalaiyaThiran/images

```
pwd
```

    '/content/drive/MyDrive/IBM_NalaiyaThiran/images'

```
!unzip flowers.zip
```

    Archive: flowers.zip
      inflating: images/f1.jpg
      inflating: images/f2.jpg
      inflating: images/f3.jpg
      inflating: images/f4.jpg
      inflating: images/f5.jpg

Image Augmentation

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

```
train_datagen = ImageDataGenerator(rescale = 1./255,zoom_range= 0.3,horizontal_flip=True,vertical_flip=True)
```

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


x_train = train_datagen.flow_from_directory(r"/content/drive/MyDrive/IBM_NalaiyaThiran/images",target_size= (64,64),class_mode= "cat

    Found 5 images belonging to 1 classes.


x_test = test_datagen.flow_from_directory(r"/content/drive/MyDrive/IBM_NalaiyaThiran/images",target_size= (64,64),class_mode= "catego

    Found 5 images belonging to 1 classes.


x_train.class_indices

    {'images': 0}


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


model = Sequential()


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


model.add(MaxPooling2D(strides=(1, 1)))


model.add(Flatten())


model.summary()

    Model: "sequential"
    _____
     Layer (type)              Output Shape              Param #
    ===============================================================
     conv2d (Conv2D)           (None, 62, 62, 32)        896
```

```
 max_pooling2d (MaxPooling2D  (None, 61, 61, 32)         0
 )

 flatten (Flatten)            (None, 119072)             0

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

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

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

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

```python
len(x_train)
```

```
    1
```

```python
model.fit(x_train,epochs = 10,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test))
```

```
    Epoch 1/10
    1/1 [==============================] - 2s 2s/step - loss: 8.0750 - accuracy: 0.0000e+00 - val_loss: 20.8234 - val_accuracy: 0.0
    Epoch 2/10
    1/1 [==============================] - 1s 848ms/step - loss: 19.1896 - accuracy: 0.0000e+00 - val_loss: 28.4673 - val_accuracy:
    Epoch 3/10
    1/1 [==============================] - 1s 634ms/step - loss: 25.9591 - accuracy: 0.0000e+00 - val_loss: 42.4799 - val_accuracy:
    Epoch 4/10
    1/1 [==============================] - 1s 684ms/step - loss: 37.1618 - accuracy: 0.0000e+00 - val_loss: 47.2677 - val_accuracy:
    Epoch 5/10
    1/1 [==============================] - 1s 648ms/step - loss: 45.7534 - accuracy: 0.0000e+00 - val_loss: 51.9293 - val_accuracy:
```

```
Epoch 6/10
1/1 [==============================] - 1s 717ms/step - loss: 51.0597 - accuracy: 0.0000e+00 - val_loss: 45.2593 - val_accuracy:
Epoch 7/10
1/1 [==============================] - 1s 654ms/step - loss: 44.8140 - accuracy: 0.0000e+00 - val_loss: 50.0706 - val_accuracy:
Epoch 8/10
1/1 [==============================] - 1s 718ms/step - loss: 50.1203 - accuracy: 0.0000e+00 - val_loss: 47.5019 - val_accuracy:
Epoch 9/10
1/1 [==============================] - 1s 677ms/step - loss: 45.3977 - accuracy: 0.0000e+00 - val_loss: 81.6582 - val_accuracy:
Epoch 10/10
1/1 [==============================] - 1s 748ms/step - loss: 78.7049 - accuracy: 0.0000e+00 - val_loss: 93.6470 - val_accuracy:
<keras.callbacks.History at 0x7fa0f6b393d0>
```

```python
model.save("flower.h5")
```

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

```python
model = load_model("flower.h5")
```

```python
img = image.load_img(r"/content/drive/MyDrive/IBM_NalaiyaThiran/images/f1.jpg")
```

```python
img
```

```
img = image.load_img(r"/content/drive/MyDrive/IBM_NalaiyaThiran/images/f1.jpg",target_size=(64,64))
```

```
img
```



```
x = image.img_to_array(img)
```

```
x
```

```
array([[[ 44.,  41.,   6.],
        [ 37.,  37.,   3.],
        [ 30.,  36.,   2.],
        ...,
        [ 39.,  40.,   9.],
        [ 84.,  71.,  29.],
        [116.,  92.,  56.]],

       [[ 49.,  45.,   8.],
        [ 42.,  39.,   4.],
```

```
       [ 27.,   32.,    2.],
       ...,
       [ 47.,   49.,    2.],
       [147.,  137.,   88.],
       [ 50.,   43.,   14.]],

      [[ 56.,   53.,   12.],
       [ 38.,   42.,    9.],
       [ 27.,   32.,    2.],
       ...,
       [ 41.,   45.,   12.],
       [ 26.,   32.,    6.],
       [ 26.,   23.,    6.]],


      ...,

      [[ 45.,   28.,    8.],
       [ 50.,   32.,   10.],
       [ 55.,   43.,   17.],
       ...,
       [ 42.,   46.,   21.],
       [  4.,    7.,    0.],
       [  6.,    9.,    2.]],

      [[ 38.,   20.,    6.],
       [ 37.,   21.,    8.],
       [ 40.,   29.,    7.],
       ...,
       [ 11.,   14.,    0.],
       [  8.,   11.,    0.],
       [  7.,   10.,    0.]],

      [[ 26.,   16.,    4.],
       [ 28.,   16.,    4.],
       [ 29.,   17.,    1.],
       ...,
       [ 12.,   14.,    3.],
       [  9.,   12.,    1.],
       [  9.,   12.,    1.]]], dtype=float32)
```

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


x
```

```
array([[[[ 44.,   41.,    6.],
         [ 37.,   37.,    3.],
         [ 30.,   36.,    2.],
         ...,
         [ 39.,   40.,    9.],
         [ 84.,   71.,   29.],
         [116.,   92.,   56.]],

        [[ 49.,   45.,    8.],
         [ 42.,   39.,    4.],
         [ 27.,   32.,    2.],
         ...,
         [ 47.,   49.,    2.],
         [147.,  137.,   88.],
         [ 50.,   43.,   14.]],

        [[ 56.,   53.,   12.],
         [ 38.,   42.,    9.],
         [ 27.,   32.,    2.],
         ...,
         [ 41.,   45.,   12.],
         [ 26.,   32.,    6.],
         [ 26.,   23.,    6.]],

        ...,

        [[ 45.,   28.,    8.],
         [ 50.,   32.,   10.],
         [ 55.,   43.,   17.],
         ...,
         [ 42.,   46.,   21.],
         [  4.,    7.,    0.],
         [  6.,    9.,    2.]],

        [[ 38.,   20.,    6.],
         [ 37.,   21.,    8.],
```

```
         [ 40.,  29.,   7.],
         ...,
         [ 11.,  14.,   0.],
         [  8.,  11.,   0.],
         [  7.,  10.,   0.]],

        [[ 26.,  16.,   4.],
         [ 28.,  16.,   4.],
         [ 29.,  17.,   1.],
         ...,
         [ 12.,  14.,   3.],
         [  9.,  12.,   1.],
         [  9.,  12.,   1.]]]], dtype=float32)
```

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

```
    1/1 [==============================] - 0s 184ms/step
```

```
pred
```

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

```
x_test.class_indices
```

```
index = ["","images"]
```

```
img = image.load_img(r"/content/drive/MyDrive/IBM_NalaiyaThiran/images/f1.jpg",target_size=(64,64))
```

```
img
```

```
img = image.load_img(r"/content/drive/MyDrive/IBM_NalaiyaThiran/images/f4.jpg")
```

```
img
```

Colab paid products - Cancel contracts here

✓ 0s   completed at 3:55 PM   ● ✕