Import Dataset from Drive and Unzip it

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

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

```
  inflating: flowers/tulip/8521597402_4b6169ba05.jpg
  inflating: flowers/tulip/8523133474_d2c0845b54.jpg
  inflating: flowers/tulip/8554190977_37ac747799_m.jpg
  inflating: flowers/tulip/8555123165_2fe57eff4f.jpg
  inflating: flowers/tulip/8562853756_73778dac25_n.jpg
  inflating: flowers/tulip/8572847041_d0cc07861f_n.jpg
  inflating: flowers/tulip/8585101979_4398146bf1_n.jpg
  inflating: flowers/tulip/8585102511_fc452e6700_n.jpg
  inflating: flowers/tulip/8585102913_d80d3e2ff7_n.jpg
  inflating: flowers/tulip/8585103457_d64697c3cf_n.jpg
  inflating: flowers/tulip/8586204750_2891bd2ec9_n.jpg
  inflating: flowers/tulip/8586205168_8294e67195_n.jpg
  inflating: flowers/tulip/8586205446_8953a6c70e_n.jpg
  inflating: flowers/tulip/8586205946_cda045f3f8_n.jpg
  inflating: flowers/tulip/8601596054_33e40c2a7a.jpg
  inflating: flowers/tulip/8603340662_0779bd87fd.jpg
  inflating: flowers/tulip/8605564823_7a59d3d92a.jpg
  inflating: flowers/tulip/8614237582_74417799f4_m.jpg
  inflating: flowers/tulip/8619064872_dea79a9eb9.jpg
  inflating: flowers/tulip/8622237974_b362574785_n.jpg
  inflating: flowers/tulip/8623170936_83f4152431.jpg
  inflating: flowers/tulip/8623173256_3f0eb4c506.jpg
  inflating: flowers/tulip/8628453641_6f87755815_m.jpg
  inflating: flowers/tulip/8659691170_09db83d023.jpg
  inflating: flowers/tulip/8668973377_c69527db42_m.jpg
  inflating: flowers/tulip/8668974855_8389ecbdca_m.jpg
  inflating: flowers/tulip/8669794378_97dda6036f_n.jpg
  inflating: flowers/tulip/8673412732_f8fd690ee4_n.jpg
  inflating: flowers/tulip/8673416166_620fc18e2f_n.jpg
  inflating: flowers/tulip/8673416556_639f5c88f1_n.jpg
  inflating: flowers/tulip/8677713853_1312f65e71.jpg
  inflating: flowers/tulip/8681825637_837a63513a_n.jpg
  inflating: flowers/tulip/8686013485_3c4dfbfd1f_n.jpg
  inflating: flowers/tulip/8686332852_c6dcb2e86b.jpg
  inflating: flowers/tulip/8687675254_c93f50d8b0_m.jpg
  inflating: flowers/tulip/8688502760_1c8d6de921_m.jpg
  inflating: flowers/tulip/8689672277_b289909f97_n.jpg
  inflating: flowers/tulip/8690789564_394eb04982_n.jpg
  inflating: flowers/tulip/8690791226_b1f015259f_n.jpg
  inflating: flowers/tulip/8695367666_0809529eaf_n.jpg
  inflating: flowers/tulip/8695372372_302135aeb2.jpg
  inflating: flowers/tulip/8697784345_e75913d220.jpg

  inflating: flowers/tulip/8702982836_75222725d7.jpg
  inflating: flowers/tulip/8706523526_a0f161b72b.jpg
  inflating: flowers/tulip/8708209606_d3aede4801.jpg
  inflating: flowers/tulip/8708856019_f3be2353a4_n.jpg
  inflating: flowers/tulip/8710148289_6fc196a0f8_n.jpg
  inflating: flowers/tulip/8711277462_b43df5454b_m.jpg
  inflating: flowers/tulip/8712230357_1298b8513b.jpg
  inflating: flowers/tulip/8712243901_54d686319e_m.jpg
  inflating: flowers/tulip/8712244311_da8e90bf8e_n.jpg
  inflating: flowers/tulip/8712260079_c0ff42e0e2_n.jpg
  inflating: flowers/tulip/8712263493_3db76c5f82.jpg
  inflating: flowers/tulip/8712266605_3787e346cd_n.jpg
  inflating: flowers/tulip/8712267391_c756f18ee7_n.jpg
  inflating: flowers/tulip/8712267813_f7a9be2ec5.jpg
  inflating: flowers/tulip/8712268519_f4c2c39a06_n.jpg
```

Data Augmentation

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

train_gen = ImageDataGenerator(rescale=1./255,
                               zoom_range=0.2,
                               horizontal_flip=True)
test_gen = ImageDataGenerator(rescale=1./255)
```

```
xtrain = train_gen.flow_from_directory('/content/flowers',
                                       target_size=(64,64),
                                       class_mode='categorical',
                                       batch_size=100)
```

```
Found 4317 images belonging to 5 classes.
```

Train

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Dense, Flatten
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

```
model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())

model.add(Dense(400,activation='relu'))
model.add(Dense(200,activation='relu'))
model.add(Dense(100,activation='relu'))
model.add(Dense(5,activation='softmax'))
```

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

```
early_stopping = EarlyStopping(monitor='accuracy',
                        patience=3)
reduce_lr = ReduceLROnPlateau(monitor='accuracy',
                        patience=5,
                        factor=0.5,min_lr=0.00001)

callback = [reduce_lr,early_stopping]



model.fit_generator(xtrain,
                    steps_per_epoch = len(xtrain),
                    callbacks=callback,
                    epochs=100)
```

```
Epoch 9/100
44/44 [==============================] - 27s 616ms/step - loss: 0.7767 - accuracy:
Epoch 10/100
44/44 [==============================] - 27s 616ms/step - loss: 0.7265 - accuracy:
Epoch 11/100
44/44 [==============================] - 29s 662ms/step - loss: 0.7027 - accuracy:
Epoch 12/100
44/44 [==============================] - 27s 614ms/step - loss: 0.6503 - accuracy:
Epoch 13/100
44/44 [==============================] - 27s 615ms/step - loss: 0.6306 - accuracy:
Epoch 14/100
44/44 [==============================] - 27s 615ms/step - loss: 0.5722 - accuracy:
Epoch 15/100
44/44 [==============================] - 27s 614ms/step - loss: 0.5580 - accuracy:
Epoch 16/100
44/44 [==============================] - 29s 654ms/step - loss: 0.5490 - accuracy:
Epoch 17/100
44/44 [==============================] - 27s 618ms/step - loss: 0.5037 - accuracy:
Epoch 18/100
44/44 [==============================] - 27s 617ms/step - loss: 0.4643 - accuracy:
Epoch 19/100
44/44 [==============================] - 27s 614ms/step - loss: 0.4619 - accuracy:
Epoch 20/100
44/44 [==============================] - 27s 618ms/step - loss: 0.4153 - accuracy:
Epoch 21/100
44/44 [==============================] - 29s 658ms/step - loss: 0.3803 - accuracy:
Epoch 22/100

44/44 [==============================] - 27s 613ms/step - loss: 0.3659 - accuracy:
Epoch 23/100
44/44 [==============================] - 27s 614ms/step - loss: 0.3458 - accuracy:
Epoch 24/100
44/44 [==============================] - 27s 615ms/step - loss: 0.3449 - accuracy:
Epoch 25/100
44/44 [==============================] - 27s 613ms/step - loss: 0.3288 - accuracy:
Epoch 26/100
44/44 [==============================] - 29s 659ms/step - loss: 0.3370 - accuracy:
Epoch 27/100
44/44 [==============================] - 27s 614ms/step - loss: 0.2959 - accuracy:
Epoch 28/100
44/44 [==============================] - 27s 613ms/step - loss: 0.2518 - accuracy:
Epoch 29/100
44/44 [==============================] - 27s 611ms/step - loss: 0.2707 - accuracy:
Epoch 30/100
44/44 [==============================] - 27s 614ms/step - loss: 0.2498 - accuracy:
Epoch 31/100
44/44 [==============================] - 27s 613ms/step - loss: 0.2335 - accuracy:
```

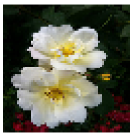
```
44/44 [==============================] - 27s 613ms/step - loss: 0.2335 - accuracy:
Epoch 32/100
44/44 [==============================] - 27s 612ms/step - loss: 0.2134 - accuracy:
Epoch 33/100
44/44 [==============================] - 27s 613ms/step - loss: 0.1918 - accuracy:
Epoch 34/100
44/44 [==============================] - 27s 613ms/step - loss: 0.2038 - accuracy:
Epoch 35/100
44/44 [==============================] - 27s 610ms/step - loss: 0.1908 - accuracy:
Epoch 36/100
44/44 [==============================] - 27s 614ms/step - loss: 0.2035 - accuracy:
<keras.callbacks.History at 0x7facb3ae6350>
```

```
model.save('flower.h5')
```

Test The Model

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

```
img = image.load_img('/content/flowers/rose/10090824183_d02c613f10_m.jpg',target_size=(64,6
img
```



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

```
array([[[14., 22.,  7.],
        [11., 22.,  6.],
        [ 8., 19.,  3.],
        ...,
        [32., 47., 24.],
        [30., 48., 22.],
        [33., 49., 23.]],

       [[13., 20., 12.],
        [11., 21., 10.],
        [11., 22.,  8.],
        ...,
        [37., 51., 26.],
        [35., 49., 26.],
        [25., 45., 20.]],

       [[19., 30., 16.],
        [19., 31., 17.],
        [16., 29., 12.],
        ...,
        [31., 47., 20.],
        [28., 49., 18.],
        [27., 43., 17.]],
       ...,

       [[15., 17.,  6.],
        [ 2.,  9.,  2.],
        [ 2.,  9.,  1.],
        ...,
        [ 8., 21., 11.],
        [ 2., 12.,  3.],
        [ 9., 16.,  9.]],

       [[12., 20.,  9.],
        [ 1.,  8.,  1.],
        [ 5., 10.,  3.],
        ...,
        [ 3.,  8.,  2.],
        [ 6., 16.,  5.],
        [ 5.,  7.,  4.]],

       [[24., 27., 18.],
        [11., 21., 13.],
        [ 8., 13.,  6.],
        ...,
        [ 1.,  6.,  0.],
        [ 2.,  9.,  1.],
        [ 2.,  9.,  1.]]], dtype=float32)
```

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

```
     array([[[[14., 22.,  7.],
              [11., 22.,  6.],
              [ 8., 19.,  3.],
              ...,
              [32., 47., 24.],
              [30., 48., 22.],
              [33., 49., 23.]],

             [[13., 20., 12.],
              [11., 21., 10.],
              [11., 22.,  8.],
              ...,
              [37., 51., 26.],
              [35., 49., 26.],
              [25., 45., 20.]],

             [[19., 30., 16.],
              [19., 31., 17.],
              [16., 29., 12.],
              ...,
              [31., 47., 20.],
              [28., 49., 18.],
              [27., 43., 17.]],

             ...,

             [[15., 17.,  6.],
              [ 2.,  9.,  2.],
              [ 2.,  9.,  1.],
              ...,
              [ 8., 21., 11.],
              [ 2., 12.,  3.],
              [ 9., 16.,  9.]],

             [[12., 20.,  9.],
              [ 1.,  8.,  1.],
              [ 5., 10.,  3.],
              ...,
              [ 3.,  8.,  2.],
              [ 6., 16.,  5.],
              [ 5.,  7.,  4.]],

             [[24., 27., 18.],
              [11., 21., 13.],
              [ 8., 13.,  6.],
              ...,
              [ 1.,  6.,  0.],
              [ 2.,  9.,  1.],
              [ 2.,  9.,  1.]]]], dtype=float32)
```

```
val = list(xtrain.class_indices.keys())
val
```

```
['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
```

```
val[np.argmax(model.predict(h))]
```

```
'daisy'
```