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

```
Archive:  /content/Flowers-Dataset.zip
replace flowers/daisy/100080576_f52e8ee070_n.jpg? [y]es, [n]o, [A]ll,
[N]one, [r]ename:
```

```
#Image Augmentation

from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.2,
horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
```

```
#Passing training and testing data to variables

xtrain =
train_datagen.flow_from_directory('/content/flowers',target_size=(64,6
4),class_mode='categorical',batch_size=100)
```

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

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

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

```
#Creating model
#importing libraries

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

```
#Adding layers

model = Sequential()
#Convolution layer
model.add(Convolution2D(32,
(3,3),activation='relu',input_shape=(64,64,3)))
#Pooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
#Flatten
model.add(Flatten())
#Fully connected layers
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
```

```
#Compile the model
```

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

```
#Fit the model
```

```
model.fit_generator(xtrain, steps_per_epoch=20, validation_data=xtest,
validation_steps=len(xtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  This is separate from the ipykernel package so we can avoid doing
imports until
```

```
20/20 [==============================] - 30s 2s/step - loss: 2.2688 -
accuracy: 0.2815 - val_loss: 1.3726 - val_accuracy: 0.4077
```

```
<keras.callbacks.History at 0x7f3694cc6550>
```

```
#Save the model
```

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

```
#Test the model
```

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

```
img=
image.load_img('/content/flowers/dandelion/11775820493_10fedf4bff_n.jp
g',target_size=(64,64))
```

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

```
array([[[ 86., 125., 182.],
        [ 88., 127., 184.],
        [ 88., 127., 184.],
        ...,
        [ 72., 111., 166.],
        [ 71., 108., 163.],
        [ 69., 108., 163.]],

       [[ 87., 126., 183.],
        [ 89., 127., 189.],
        [ 90., 129., 186.],
        ...,
        [ 74., 113., 168.],
        [ 74., 111., 164.],
        [ 70., 110., 162.]],
       [[ 89., 128., 185.],
        [ 92., 128., 188.],
        [ 91., 130., 187.],
        ...,
        [ 75., 115., 167.],
        [ 74., 114., 166.],
        [ 72., 111., 166.]],

       ...,

       [[ 40.,  64.,   2.],
        [ 38.,  59.,   2.],
        [ 22.,  45.,  17.],
        ...,
        [  6.,  14.,   3.],
        [ 11.,  31.,  20.],
        [ 20.,  36.,  23.]],

       [[ 33.,  56.,   4.],
        [ 35.,  51.,   6.],
        [ 21.,  41.,  13.],
        ...,
        [  1.,  11.,   2.],
        [  0.,   8.,   0.],
        [  8.,  22.,   9.]],

       [[ 31.,  48.,   3.],
        [ 22.,  37.,   6.],
        [ 33.,  49.,   0.],
        ...,
        [  1.,  10.,   7.],
        [  3.,   8.,   2.],
        [  0.,   9.,   1.]]], dtype=float32)
```

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

#Predict
model.predict(x)
xtrain.class_indices
op= ['daisy','dandelion','rose','sunflower','tulip']
pred= np.argmax(model.predict(x))
op[pred]
```

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

```
#Model tuning
#import libraries
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
early_stopping = EarlyStopping(monitor='value_accuracy',patience=5)
```

```
reduce_lr = ReduceLROnPlateau(monitor='value_accuracy',patience=5,
factor=0.5, min_lr=0.0001)
callback= [reduce_lr, early_stopping]
```

```
model.fit_generator(xtrain, steps_per_epoch=100,callbacks= callback,
validation_data=xtest, validation_steps=len(xtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
UserWarning: `Model.fit_generator` is deprecated and will be removed
in a future version. Please use `Model.fit`, which supports
generators.
  """Entry point for launching an IPython kernel.
```

```
 44/100 [============>.................] - ETA: 37s - loss: 0.7350 -
accuracy: 0.7227
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training.
Make sure that your dataset or generator can generate at least
`steps_per_epoch * epochs` batches (in this case, 100 batches). You
may need to use the repeat() function when building your dataset.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`value_accuracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
WARNING:tensorflow:Early stopping conditioned on metric
`value_accuracy` which is not available. Available metrics are:
loss,accuracy,val_loss,val_accuracy,lr
```

```
 100/100 [==============================] - 45s 446ms/step - loss:
0.7350 - accuracy: 0.7227 - val_loss: 0.6672 - val_accuracy: 0.7433 -
lr: 0.0010
```

```
<keras.callbacks.History at 0x7f3694a2b390>
```

```
#Test the model
```

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

```
img=
image.load_img('/content/flowers/rose/10090824183_d02c613f10_m.jpg',ta
rget_size=(64,64))
```

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

```
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)
```

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

#Predict
model.predict(x)
xtrain.class_indices
op= ['daisy','dandelion','rose','sunflower','tulip']
```

```
pred= np.argmax(model.predict(x))
op[pred]
```

{"type":"string"}