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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

tain_data=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,vertical
_flip=True)
test_data=ImageDataGenerator(rescale=1./255)

x_train=train_data.flow_from_directory(r"E:\assignment3\dataset\Training",target_size=(64,64),batch_size=32,
class_mode="categorical")
x_test=test_data.flow_from_directory(r"E:\assignment3\dataset\Testing",target_size=(64,64),batch_size=32,clas
s_mode="categorical")
```

```
Found 3453 images belonging to 5 classes.
Found 864 images belonging to 5 classes.
```

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

```
model=Sequential()

model.add(Convolution2D(64,(3,3),input_shape=(64,64,3),activation='relu'))

model.add(MaxPooling2D(pool_size=(3,3)))

model.add(Flatten())

model.add(Dense(units=5,kernel_initializer="random_uniform",activation="softmax"))

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

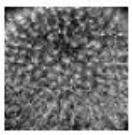
I

```
model.fit_generator(x_train,steps_per_epoch=108,epochs=30,validation_data=x_test,validation_steps=27)
```

```
Epoch 1/30
108/108 [==============================] - 88s 798ms/step - loss: 1.6050 - accuracy: 0.2299 - val_lo
ss: 1.6017 - val_accuracy: 0.2431
Epoch 2/30
108/108 [==============================] - 70s 646ms/step - loss: 1.6005 - accuracy: 0.2438 - val_lo
ss: 1.5994 - val_accuracy: 0.2431
Epoch 3/30
108/108 [==============================] - 71s 654ms/step - loss: 1.5992 - accuracy: 0.2438 - val_lo
ss: 1.5989 - val_accuracy: 0.2431
Epoch 4/30
108/108 [==============================] - 70s 648ms/step - loss: 1.5988 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 5/30
```

108/108 [==============================] - 74s 685ms/step - loss: 1.5988 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 6/30
108/108 [==============================] - 77s 712ms/step - loss: 1.5986 - accuracy: 0.2438 - val_loss: 1.5985 - val_accuracy: 0.2431
Epoch 7/30
108/108 [==============================] - 80s 739ms/step - loss: 1.5985 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 8/30
108/108 [==============================] - 76s 702ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 9/30
108/108 [==============================] - 75s 692ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 10/30
108/108 [==============================] - 78s 721ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 11/30
108/108 [==============================] - 75s 693ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 12/30
108/108 [==============================] - 75s 695ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 13/30
108/108 [==============================] - 81s 745ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 14/30
108/108 [==============================] - 76s 700ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 15/30
108/108 [==============================] - 80s 742ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 16/30
108/108 [==============================] - 75s 693ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 17/30
108/108 [==============================] - 75s 692ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 18/30
108/108 [==============================] - 75s 696ms/step - loss: 1.5986 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 19/30
108/108 [==============================] - 75s 694ms/step - loss: 1.5986 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 20/30
108/108 [==============================] - 75s 694ms/step - loss: 1.5988 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 21/30
108/108 [==============================] - 75s 690ms/step - loss: 1.5986 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 22/30
108/108 [==============================] - 76s 702ms/step - loss: 1.5986 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 23/30
108/108 [==============================] - 76s 705ms/step - loss: 1.5987 - accuracy: 0.2438 - val_loss: 1.5987 - val_accuracy: 0.2431
Epoch 24/30

```
108/108 [==============================] - 75s 692ms/step - loss: 1.5987 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 25/30
108/108 [==============================] - 75s 690ms/step - loss: 1.5988 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 26/30
108/108 [==============================] - 75s 695ms/step - loss: 1.5986 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 27/30
108/108 [==============================] - 76s 699ms/step - loss: 1.5987 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 28/30
108/108 [==============================] - 75s 692ms/step - loss: 1.5987 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 29/30
108/108 [==============================] - 75s 690ms/step - loss: 1.5987 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431
Epoch 30/30
108/108 [==============================] - 75s 696ms/step - loss: 1.5988 - accuracy: 0.2438 - val_lo
ss: 1.5987 - val_accuracy: 0.2431

<keras.callbacks.History at 0x23bcb0e0d60>

model.save("flower.h5")

from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

import numpy as np

model=load_model("flower.h5")

img=image.load_img("dandelion.jpg",target_size=(64,64))

img
```



```
type(img)

PIL.Image.Image
x=image.img_to_array(img)

x

array([[[ 4.,  4.,  4.],
        [ 11.,  11.,  11.],
        [ 9.,  9.,  9.],
        ...,
        [ 11.,  11.,  11.],
        [ 9.,  9.,  9.],
```

```
       [ 10.,  10.,  10.]],

      [[ 12.,  12.,  12.],
       [ 21.,  21.,  21.],
       [ 19.,  19.,  19.],
       ...,
       [ 15.,  15.,  15.],
       [ 23.,  23.,  23.],
       [  2.,   2.,   2.]],

      [[ 23.,  23.,  23.],
       [ 52.,  52.,  52.],
       [ 41.,  41.,  41.],
       ...,
       [ 49.,  49.,  49.],
       [ 34.,  34.,  34.],
       [ 42.,  42.,  42.]],

      ...,

      [[ 56.,  56.,  56.],
       [121., 121., 121.],
       [ 44.,  44.,  44.],
       ...,
       [113., 113., 113.],
       [ 24.,  24.,  24.],
       [ 12.,  12.,  12.]],

      [[ 54.,  54.,  54.],
       [ 30.,  30.,  30.],
       [ 36.,  36.,  36.],
       ...,
       [102., 102., 102.],
       [ 42.,  42.,  42.],
       [ 10.,  10.,  10.]],

      [[ 23.,  23.,  23.],
       [ 20.,  20.,  20.],
       [ 11.,  11.,  11.],
       ...,
       [ 77.,  77.,  77.],
       [  3.,   3.,   3.],
       [ 10.,  10.,  10.]]], dtype=float32)
```

```
x.shape
```

```
(64, 64, 3)
```

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

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

```
1/1 [==============================] - 0s 63ms/step
```

```
pred_prob
```

```
array([[0.17660806, 0.24450503, 0.18085377, 0.17052399, 0.22750916]],
      dtype=float32)

class_name=["Daisy","Dandelion","Rose","Sunflower","Tulip"]
pred_id=pred_prob.argmax(axis=1)[0]


pred_id


1

print("Predicted animalis ",str(class_name[pred_id]))
Predicted animalis  Dandelion
```