```
from google.colab import drive

drive.mount('/content/drive')
```

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

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

train_datagen =
ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,
vertical_flip=False,validation_split=0.2)

test_datagen = ImageDataGenerator(rescale=1./255,validation_split=0.2)

#to split data into train and test add validation_split and subset
x_train=train_datagen.flow_from_directory(r"/content/drive/MyDrive/
flowers",target_size=(64,64),class_mode='categorical',batch_size=100,s
ubset = 'training')
```

```
Found 3457 images belonging to 5 classes.
```

```
x_test=test_datagen.flow_from_directory(r"/content/drive/MyDrive/
flowers",target_size=(64,64),class_mode='categorical',batch_size=100,s
ubset = 'validation')
```

```
Found 860 images belonging to 5 classes.
```

```
x_train.class_indices
```

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

```
from tensorflow.keras.models import Sequential

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

model=Sequential()

model.add(Convolution2D(32,
(3,3),input_shape=(64,64,3),activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 62, 62, 32)        896
```

```
 max_pooling2d_6 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 flatten_6 (Flatten)         (None, 30752)             0

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

```python
#hidden layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(75,activation='relu'))
model.add(Dense(5,activation='softmax'))#op layer
```

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

```python
len(x_train)
```

```
35
```

```python
3457/100
```

```
34.57
```

```python
len(x_test)
```

```
9
```

```python
model.fit(x_train,steps_per_epoch=len(x_train),validation_data=x_test,
validation_steps=len(x_test),epochs=30)
```

```
Epoch 1/30
35/35 [==============================] - 33s 950ms/step - loss: 0.4863
- accuracy: 0.8261 - val_loss: 1.3241 - val_accuracy: 0.5930
Epoch 2/30
35/35 [==============================] - 33s 932ms/step - loss: 0.4793
- accuracy: 0.8230 - val_loss: 1.3508 - val_accuracy: 0.6163
Epoch 3/30
35/35 [==============================] - 33s 926ms/step - loss: 0.4529
- accuracy: 0.8241 - val_loss: 1.3197 - val_accuracy: 0.5860
Epoch 4/30
35/35 [==============================] - 32s 925ms/step - loss: 0.4372
- accuracy: 0.8426 - val_loss: 1.2772 - val_accuracy: 0.6326
Epoch 5/30
35/35 [==============================] - 32s 919ms/step - loss: 0.3831
- accuracy: 0.8568 - val_loss: 1.5694 - val_accuracy: 0.5849
Epoch 6/30
```

```
35/35 [==============================] - 32s 919ms/step - loss: 0.3878
- accuracy: 0.8597 - val_loss: 1.2102 - val_accuracy: 0.6488
Epoch 7/30
35/35 [==============================] - 33s 928ms/step - loss: 0.3470
- accuracy: 0.8805 - val_loss: 1.3690 - val_accuracy: 0.6233
Epoch 8/30
35/35 [==============================] - 32s 922ms/step - loss: 0.3300
- accuracy: 0.8872 - val_loss: 1.3737 - val_accuracy: 0.6360
Epoch 9/30
35/35 [==============================] - 33s 937ms/step - loss: 0.3249
- accuracy: 0.8823 - val_loss: 1.5146 - val_accuracy: 0.6279
Epoch 10/30
35/35 [==============================] - 33s 926ms/step - loss: 0.3242
- accuracy: 0.8794 - val_loss: 1.4434 - val_accuracy: 0.6326
Epoch 11/30
35/35 [==============================] - 33s 934ms/step - loss: 0.2771
- accuracy: 0.9043 - val_loss: 1.3744 - val_accuracy: 0.6477
Epoch 12/30
35/35 [==============================] - 33s 950ms/step - loss: 0.2410
- accuracy: 0.9135 - val_loss: 1.7165 - val_accuracy: 0.6302
Epoch 13/30
35/35 [==============================] - 33s 944ms/step - loss: 0.2448
- accuracy: 0.9121 - val_loss: 1.5927 - val_accuracy: 0.6093
Epoch 14/30
35/35 [==============================] - 36s 1s/step - loss: 0.2700 -
accuracy: 0.9077 - val_loss: 1.5080 - val_accuracy: 0.6337
Epoch 15/30
35/35 [==============================] - 34s 960ms/step - loss: 0.2475
- accuracy: 0.9092 - val_loss: 1.5869 - val_accuracy: 0.6337
Epoch 16/30
35/35 [==============================] - 34s 961ms/step - loss: 0.2255
- accuracy: 0.9251 - val_loss: 1.5303 - val_accuracy: 0.6581
Epoch 17/30
35/35 [==============================] - 34s 969ms/step - loss: 0.1683
- accuracy: 0.9439 - val_loss: 1.8287 - val_accuracy: 0.6209
Epoch 18/30
35/35 [==============================] - 34s 957ms/step - loss: 0.2143
- accuracy: 0.9225 - val_loss: 1.7494 - val_accuracy: 0.6279
Epoch 19/30
35/35 [==============================] - 34s 968ms/step - loss: 0.1827
- accuracy: 0.9340 - val_loss: 1.8444 - val_accuracy: 0.6116
Epoch 20/30
35/35 [==============================] - 34s 963ms/step - loss: 0.2180
- accuracy: 0.9300 - val_loss: 1.7823 - val_accuracy: 0.5977
Epoch 21/30
35/35 [==============================] - 34s 954ms/step - loss: 0.1630
- accuracy: 0.9459 - val_loss: 1.6510 - val_accuracy: 0.6500
Epoch 22/30
35/35 [==============================] - 34s 956ms/step - loss: 0.1381
- accuracy: 0.9543 - val_loss: 1.6596 - val_accuracy: 0.6349
```

```
Epoch 23/30
35/35 [==============================] - 34s 964ms/step - loss: 0.1241
- accuracy: 0.9581 - val_loss: 2.1146 - val_accuracy: 0.6209
Epoch 24/30
35/35 [==============================] - 35s 995ms/step - loss: 0.1484
- accuracy: 0.9479 - val_loss: 1.8599 - val_accuracy: 0.6407
Epoch 25/30
35/35 [==============================] - 35s 999ms/step - loss: 0.1400
- accuracy: 0.9531 - val_loss: 1.8035 - val_accuracy: 0.6233
Epoch 26/30
35/35 [==============================] - 35s 985ms/step - loss: 0.1397
- accuracy: 0.9537 - val_loss: 2.0482 - val_accuracy: 0.6023
Epoch 27/30
35/35 [==============================] - 34s 969ms/step - loss: 0.1177
- accuracy: 0.9589 - val_loss: 1.9821 - val_accuracy: 0.6023
Epoch 28/30
35/35 [==============================] - 34s 957ms/step - loss: 0.0901
- accuracy: 0.9734 - val_loss: 2.3850 - val_accuracy: 0.6093
Epoch 29/30
35/35 [==============================] - 33s 943ms/step - loss: 0.1163
- accuracy: 0.9624 - val_loss: 2.2573 - val_accuracy: 0.6244
Epoch 30/30
35/35 [==============================] - 34s 968ms/step - loss: 0.1102
- accuracy: 0.9633 - val_loss: 2.1816 - val_accuracy: 0.6267

<keras.callbacks.History at 0x7f5dfca37b90>
```

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

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

```
model=load_model('flowers.h5')
```

```
img =
image.load_img(r"/content/drive/MyDrive/flowers/sunflower/9904127656_f
76a5a4811_m.jpg")
```

```
img
```


```
img =
image.load_img(r"/content/drive/MyDrive/flowers/sunflower/9904127656_f
76a5a4811_m.jpg",target_size=(64,64))
```

```
img
```

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

```
x
```

```
array([[[ 94.,  61.,  30.],
        [137.,  98.,  57.],
        [141., 105.,  57.],
        ...,
        [181., 174., 119.],
        [186., 173., 120.],
        [185., 174., 118.]],

       [[130.,  81.,  38.],
        [155., 108.,  62.],
        [143., 105.,  56.],
        ...,
        [177., 171., 111.],
        [184., 174., 115.],
        [185., 168., 112.]],
```

```
        [[157., 107.,  56.],
         [152., 106.,  57.],
         [148., 107.,  53.],
         ...,
         [116.,  65.,  22.],
         [ 87.,  52.,   0.],
         [114.,  90.,  28.]],

        ...,

        [[245., 244., 249.],
         [244., 245., 249.],
         [218., 222., 225.],
         ...,
         [171., 142., 100.],
         [172., 144.,  94.],
         [165., 137.,  89.]],

        [[246., 248., 247.],
         [241., 242., 244.],
         [207., 207., 205.],
         ...,
         [173., 144., 100.],
         [176., 148., 100.],
         [171., 143.,  96.]],

        [[249., 249., 251.],
         [244., 248., 249.],
         [203., 191., 177.],
         ...,
         [164., 136.,  86.],
         [168., 140.,  93.],
         [169., 140.,  98.]]], dtype=float32)
```

```
x.shape
```

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

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

```
(1, 64, 64, 3)
```

```
y=np.argmax(model.predict(x),axis=1)
```

```
y
```

```
array([3])
```

```
x_train.class_indices
```

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

```
img =
image.load_img(r"/content/drive/MyDrive/flowers/tulip/9444202147_40529
0415b_n.jpg",target_size=(64,64))
print(img)
x=image.img_to_array(img)
print(x)
x=np.expand_dims(x,axis=0)
print(x.shape)
y=np.argmax(model.predict(x),axis=1)
print(x_train.class_indices)
print(y)
```

```
<PIL.Image.Image image mode=RGB size=64x64 at 0x7F5DFC8AEA90>
[[[ 69.  66.  57.]
  [ 71.  68.  59.]
  [ 73.  70.  63.]
  ...
  [ 36.  36.  26.]
  [ 37.  34.  25.]
  [ 36.  33.  24.]]

 [[ 75.  72.  63.]
  [ 77.  74.  65.]
  [ 82.  78.  69.]
  ...
  [ 37.  37.  25.]
  [ 40.  38.  26.]
  [ 38.  36.  24.]]

 [[ 87.  85.  73.]
  [ 91.  89.  77.]
  [ 94.  90.  78.]
  ...
  [ 46.  44.  32.]
  [ 44.  42.  30.]
  [ 41.  39.  27.]]

 ...

 [[ 76.  82.  56.]
  [ 73.  79.  53.]
  [ 68.  73.  50.]
  ...
  [122. 129. 139.]
  [128. 132. 144.]
  [128. 132. 143.]]

 [[ 74.  83.  54.]
  [ 74.  80.  54.]
  [ 69.  74.  51.]
```

```
  ...
  [122. 129. 139.]
  [124. 130. 142.]
  [124. 128. 139.]]

 [[ 77.  83.  55.]
  [ 77.  81.  56.]
  [ 71.  76.  53.]
  ...
  [122. 124. 137.]
  [121. 127. 139.]
  [121. 125. 136.]]]
(1, 64, 64, 3)
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
[4]
```