In [1]:

```
import warnings
warnings.filterwarnings('ignore')
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
```

In [2]:

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

In [ ]:

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

In [4]:

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

In [9]:

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

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

In [11]:

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

In [13]:

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

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

In [14]:

```
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.
```

Create Model

In [15]:

```
model = Sequential()
```

Convolution Layer

In [16]:

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

MaxPooling

In [17]:

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

Flatten

In [18]:

```
model.add(Flatten())
```

Dense Layer

In [19]:

```
model.add(Dense(300,activation='relu'))  #hiddenlayer 1
model.add(Dense(150,activation='relu'))  #hiddenlayer 2
```

Output

In [20]:

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

Compile the model

In [21]:

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

Fit The Model

In [22]:

```
model.fit_generator(xtrain,steps_per_epoch=108,epochs=30,validation_data=xtest,validation
_steps=27)
```

```
Epoch 1/30
 44/108 [===========>..................] - ETA: 42s - loss: 1.4033 - accuracy: 0.4267
```

```
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 ca
se, 3240 batches). You may need to use the repeat() function when building your dataset.
```

```
108/108 [==============================] - 41s 364ms/step - loss: 1.4033 - accuracy: 0.42
67 - val_loss: 1.1901 - val_accuracy: 0.5167
```

Out[22]:

```
<keras.callbacks.History at 0x7f6e148e2350>
```

Save The Model

In [23]:

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

Test The Model

In [24]:

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

In [25]:

```
img = image.load_img('/content/flowers/sunflower/10386503264_e05387e1f7_m.jpg',target_siz
e=(64,64))
```

In [26]:

```
img
```

Out[26]:

In [27]:

```
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
```

In [28]:

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

```
1/1 [==============================] - 0s 127ms/step
```

In [34]:

```
class_name=["Daisy","Dandelion","Rose","Tulip","Sunflower"]
```

In [35]:

```
pred_id=pred_prob.argmax(axis=1)[0]
pred_id
```

Out[35]:

```
4
```

In [36]:

```
print("Predicted flower is",str(class_name[pred_id]))
```

```
Predicted flower is Sunflower
```