In [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

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

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
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 [ ]:

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

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

Create Model

In [ ]:

```
model = Sequential()
```

Convolution Layer

In [ ]:

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

MaxPooling

In [ ]:

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

Flatten

In [ ]:

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

Dense Layer

In [ ]:

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

Output

In [ ]:

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

Compile the model

In [ ]:

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

Fit The Model

In [ ]:

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

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

```
Epoch 1/30
 44/108 [===========>.................] - ETA: 44s - loss: 1.5911 - accuracy: 0.3868
```

```
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 [==============================] - 42s 375ms/step - loss: 1.5911 - accuracy: 0.38
68 - val_loss: 1.2107 - val_accuracy: 0.4737
```

Out[ ]:

```
<keras.callbacks.History at 0x7f8c1fef34d0>
```

Save The Model

In [ ]:

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

Test The Model

In [ ]:

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

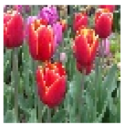
In [ ]:

```
img = image.load_img('/content/flowers/tulip/8690789564_394eb04982_n.jpg',target_size=(64
[illegible]
,64))
```

In [ ]:

```
img
```

Out[ ]:



In [ ]:

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

In [ ]:

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

```
1/1 [==============================] - 0s 23ms/step
```

In [61]:

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

In [62]:

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

Out[62]:

```
2
```

In [63]:

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

```
Predicted flower is Tulip
```