```
#Importing libraries

import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator

#Data Processing

#Training Image Processing

train_datagen = ImageDataGenerator(rescale=1./255,
                shear_range=0.2,
                zoom_range=0.2,
                horizontal_flip=True)
training_set =
train_datagen.flow_from_directory('training_set',target_size=(64,64),
batch_size=32, class_mode='categorical')

Found 3278 images belonging to 5 classes.

#Test Image processing

test_datagen= ImageDataGenerator(rescale=1./255)
test_set =
test_datagen.flow_from_directory('test_set',target_size=(64,64),
batch_size=32, class_mode='categorical')

Found 1039 images belonging to 5 classes.

#Building Model

cnn= tf.keras.models.Sequential()

#Building Convolution layer

cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='re
lu',input_shape=[64,64,3]))   #Convolution layer
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
#Pooling

cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='re
lu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))

cnn.add(tf.keras.layers.Dropout(0.5))

cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))

cnn.add(tf.keras.layers.Dense(units=5,activation='softmax'))

cnn.compile(optimizer='rmsprop',loss='categorical_crossentropy',metric
s=['accuracy'])
```

```
cnn.fit(x=training_set,validation_data=test_set,epochs=30 )

Epoch 1/30
103/103 [==============================] - 35s 331ms/step - loss:
1.4266 - accuracy: 0.3813 - val_loss: 1.1989 - val_accuracy: 0.5149
Epoch 2/30
103/103 [==============================] - 33s 317ms/step - loss:
1.1291 - accuracy: 0.5488 - val_loss: 1.0944 - val_accuracy: 0.5736
Epoch 3/30
103/103 [==============================] - 33s 323ms/step - loss:
1.0176 - accuracy: 0.6016 - val_loss: 0.9697 - val_accuracy: 0.6323
Epoch 4/30
103/103 [==============================] - 33s 323ms/step - loss:
0.9585 - accuracy: 0.6309 - val_loss: 1.0285 - val_accuracy: 0.5823
Epoch 5/30
103/103 [==============================] - 33s 323ms/step - loss:
0.8913 - accuracy: 0.6531 - val_loss: 0.9385 - val_accuracy: 0.6343
Epoch 6/30
103/103 [==============================] - 33s 324ms/step - loss:
0.8568 - accuracy: 0.6589 - val_loss: 0.8909 - val_accuracy: 0.6795
Epoch 7/30
103/103 [==============================] - 33s 323ms/step - loss:
0.8209 - accuracy: 0.6693 - val_loss: 1.1313 - val_accuracy: 0.5890
Epoch 8/30
103/103 [==============================] - 32s 313ms/step - loss:
0.7836 - accuracy: 0.6998 - val_loss: 0.8676 - val_accuracy: 0.6795
Epoch 9/30
103/103 [==============================] - 31s 300ms/step - loss:
0.7472 - accuracy: 0.7096 - val_loss: 0.9217 - val_accuracy: 0.6679
Epoch 10/30
103/103 [==============================] - 30s 296ms/step - loss:
0.7325 - accuracy: 0.7248 - val_loss: 0.9326 - val_accuracy: 0.6391
Epoch 11/30
103/103 [==============================] - 31s 300ms/step - loss:
0.7020 - accuracy: 0.7398 - val_loss: 1.0401 - val_accuracy: 0.6372
Epoch 12/30
103/103 [==============================] - 31s 304ms/step - loss:
0.6666 - accuracy: 0.7489 - val_loss: 0.9099 - val_accuracy: 0.6622
Epoch 13/30
103/103 [==============================] - 32s 307ms/step - loss:
0.6563 - accuracy: 0.7553 - val_loss: 1.1648 - val_accuracy: 0.6295
Epoch 14/30
103/103 [==============================] - 31s 304ms/step - loss:
0.6358 - accuracy: 0.7538 - val_loss: 0.9421 - val_accuracy: 0.6535
Epoch 15/30
103/103 [==============================] - 32s 310ms/step - loss:
0.6189 - accuracy: 0.7682 - val_loss: 0.8681 - val_accuracy: 0.6882
Epoch 16/30
103/103 [==============================] - 31s 301ms/step - loss:
0.5735 - accuracy: 0.7874 - val_loss: 0.9175 - val_accuracy: 0.6997
```

```
Epoch 17/30
103/103 [==============================] - 29s 281ms/step - loss:
0.5659 - accuracy: 0.7935 - val_loss: 0.8772 - val_accuracy: 0.6862
Epoch 18/30
103/103 [==============================] - 33s 315ms/step - loss:
0.5592 - accuracy: 0.7904 - val_loss: 0.8315 - val_accuracy: 0.7180
Epoch 19/30
103/103 [==============================] - 31s 302ms/step - loss:
0.5537 - accuracy: 0.7910 - val_loss: 0.7949 - val_accuracy: 0.7093
Epoch 20/30
103/103 [==============================] - 31s 301ms/step - loss:
0.5194 - accuracy: 0.8045 - val_loss: 0.8603 - val_accuracy: 0.7036
Epoch 21/30
103/103 [==============================] - 34s 330ms/step - loss:
0.5413 - accuracy: 0.8020 - val_loss: 0.9351 - val_accuracy: 0.6891
Epoch 22/30
103/103 [==============================] - 33s 322ms/step - loss:
0.4984 - accuracy: 0.8115 - val_loss: 0.8462 - val_accuracy: 0.7132
Epoch 23/30
103/103 [==============================] - 33s 321ms/step - loss:
0.4752 - accuracy: 0.8231 - val_loss: 0.9173 - val_accuracy: 0.6939
Epoch 24/30
103/103 [==============================] - 33s 320ms/step - loss:
0.4745 - accuracy: 0.8258 - val_loss: 0.8550 - val_accuracy: 0.7026
Epoch 25/30
103/103 [==============================] - 33s 319ms/step - loss:
0.4617 - accuracy: 0.8298 - val_loss: 1.0153 - val_accuracy: 0.6766
Epoch 26/30
103/103 [==============================] - 33s 316ms/step - loss:
0.4523 - accuracy: 0.8331 - val_loss: 0.8944 - val_accuracy: 0.6910
Epoch 27/30
103/103 [==============================] - 33s 318ms/step - loss:
0.4362 - accuracy: 0.8398 - val_loss: 0.9613 - val_accuracy: 0.7122
Epoch 28/30
103/103 [==============================] - 33s 320ms/step - loss:
0.4258 - accuracy: 0.8392 - val_loss: 1.0502 - val_accuracy: 0.7113
Epoch 29/30
103/103 [==============================] - 33s 320ms/step - loss:
0.4345 - accuracy: 0.8417 - val_loss: 1.1447 - val_accuracy: 0.6785
Epoch 30/30
103/103 [==============================] - 33s 316ms/step - loss:
0.4172 - accuracy: 0.8487 - val_loss: 0.9629 - val_accuracy: 0.7084

<keras.callbacks.History at 0x1b1e5757730>
```

```
#preprocess new image

from keras.preprocessing import image
test_image = tf.keras.utils.load_img('Prediction/rose1.jpeg',
target_size=(64,64))
test_image = tf.keras.utils.img_to_array(test_image)
```

```
test_image = np.expand_dims(test_image, axis=0)
result=cnn.predict(test_image)
```

```
1/1 [==============================] - 0s 35ms/step
```

```
training_set.class_indices
```

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

```
print(result)
```

```
[[0. 0. 1. 0. 0.]]
```

```
if result[0][0]==1:
    print('Daisy')
elif result[0][1]==1:
    print('Dandelion')
elif result[0][2]==1:
    print('Rose')
elif result[0][3]==1:
    print('Sunflower')
elif result[0][4]==1:
    print('Tulip')
```

```
Rose
```