Assignment -3

Python Programming

Assignment Date	4 November 2022
Student Name	P Mohamed Iburahim
Student Roll Number	212219060163
Maximum Marks	2 Marks

In [6]:
```
import tensorflow as tf
```

In [7]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
#Augmenting the input training images
```

In [11]:
```
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',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')
```

```
Found 4103 images belonging to 5 classes.
```

In [12]:
```
test_datagen = ImageDataGenerator(
        rescale=1./255)

test_data = test_datagen.flow_from_directory(
        'Testing',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')
```

```
Found 214 images belonging to 5 classes.
```

In [ ]:
```
#Building the model
```

In [13]:
```
cnn = tf.keras.models.Sequential()
```

In [ ]:
```
#Adding convolution layer
```

In [14]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation ="relu",input_shape =[64,64,3]))

cnn.add(tf.keras.layers.MaxPool2D(pool_size = 2,strides=2))
```

In [15]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation ="relu"))

cnn.add(tf.keras.layers.MaxPool2D(pool_size = 2,strides=2))
```

In [16]:
```
cnn.add(tf.keras.layers.Dropout(0.5))
```

In [ ]:
```
# Flattening the layers
```

In [17]:
```
cnn.add(tf.keras.layers.Flatten())
```

In [ ]:
```
# Adding dense layers(Hidden Layers)
```

In [18]:
```
cnn.add(tf.keras.layers.Dense(units=128 ,activation ="relu"))
```

In [19]:
```
cnn.add(tf.keras.layers.Dense(units=5,activation="softmax"))
```

In [ ]:
```
#compilation of the neural network model
```

In [20]:
```
cnn.compile(optimizer="rmsprop",loss="categorical_crossentropy" ,metrics =["accuracy"])
```

In [ ]:
```
#Fitting the neural network model and training it
```

In [41]:
```
cnn.fit(x = training_set , validation_data =test_data , epochs = 30 )
```

```
Epoch 1/30
129/129 [==============================] - 34s 254ms/step - loss: 1.3400 - accuracy: 0.4350 - val_loss: 1.0596 - val_accuracy: 0.6168
Epoch 2/30
129/129 [==============================] - 33s 253ms/step - loss: 1.0957 - accuracy: 0.5659 - val_loss: 1.1546 - val_accuracy: 0.6168
Epoch 3/30
129/129 [==============================] - 36s 279ms/step - loss: 0.9823 - accuracy: 0.6176 - val_loss: 1.0383 - val_accuracy: 0.5841
Epoch 4/30
129/129 [==============================] - 37s 285ms/step - loss: 0.9194 - accuracy: 0.6432 - val_loss: 0.8612 - val_accuracy: 0.6776
Epoch 5/30
129/129 [==============================] - 37s 289ms/step - loss: 0.8707 - accuracy: 0.6727 - val_loss: 1.1994 - val_accuracy: 0.5514
Epoch 6/30
129/129 [==============================] - 41s 315ms/step - loss: 0.8155 - accuracy: 0.6856 - val_loss: 0.9825 - val_accuracy: 0.6916
Epoch 7/30
129/129 [==============================] - 37s 285ms/step - loss: 0.7836 - accuracy: 0.7002 - val_loss: 0.9143 - val_accuracy: 0.6636
Epoch 8/30
129/129 [==============================] - 36s 280ms/step - loss: 0.7603 - accuracy: 0.7090 - val_loss: 0.8084 - val_accuracy: 0.7243
Epoch 9/30
129/129 [==============================] - 33s 257ms/step - loss: 0.7361 - accuracy: 0.7187 - val_loss: 0.8042 - val_accuracy: 0.7150
Epoch 10/30
129/129 [==============================] - 32s 250ms/step - loss: 0.6901 - accuracy: 0.7387 - val_loss: 0.9286 - val_accuracy: 0.6589
Epoch 11/30
129/129 [==============================] - 35s 273ms/step - loss: 0.6722 - accuracy: 0.7453 - val_loss: 1.0362 - val_accuracy: 0.6822
Epoch 12/30
129/129 [==============================] - 35s 270ms/step - loss: 0.6659 - accuracy: 0.7534 - val_loss: 0.7733 - val_accuracy: 0.7056
Epoch 13/30
129/129 [==============================] - 34s 261ms/step - loss: 0.6291 - accuracy: 0.7655 - val_loss: 0.8955 - val_accuracy: 0.6916
Epoch 14/30
129/129 [==============================] - 37s 284ms/step - loss: 0.6128 - accuracy: 0.7702 - val_loss: 0.9361 - val_accuracy: 0.6542
Epoch 15/30
129/129 [==============================] - 36s 279ms/step - loss: 0.5988 - accuracy: 0.7780 - val_loss: 0.8789 - val_accuracy: 0.6916
Epoch 16/30
129/129 [==============================] - 36s 281ms/step - loss: 0.5822 - accuracy: 0.7775 - val_loss: 0.9812 - val_accuracy: 0.6729
Epoch 17/30
129/129 [==============================] - 38s 298ms/step - loss: 0.5802 - accuracy: 0.7870 - val_loss: 0.8973 - val_accuracy: 0.7056
Epoch 18/30
129/129 [==============================] - 40s 306ms/step - loss: 0.5724 - accuracy: 0.7875 - val_loss: 0.8542 - val_accuracy: 0.7056
Epoch 19/30
129/129 [==============================] - 39s 305ms/step - loss: 0.5624 - accuracy: 0.7955 - val_loss: 0.7468 - val_accuracy: 0.7430
Epoch 20/30
129/129 [==============================] - 39s 303ms/step - loss: 0.5542 - accuracy: 0.7919 - val_loss: 0.8988 - val_accuracy: 0.7150
Epoch 21/30
129/129 [==============================] - 43s 329ms/step - loss: 0.5241 - accuracy: 0.8040 - val_loss: 1.0677 - val_accuracy: 0.6963
Epoch 22/30
129/129 [==============================] - 38s 296ms/step - loss: 0.5146 - accuracy: 0.8172 - val_loss: 0.8774 - val_accuracy: 0.7243
Epoch 23/30
129/129 [==============================] - 39s 302ms/step - loss: 0.5153 - accuracy: 0.8172 - val_loss: 0.8348 - val_accuracy: 0.6963
Epoch 24/30
129/129 [==============================] - 45s 348ms/step - loss: 0.5067 - accuracy: 0.8153 - val_loss: 0.9380 - val_accuracy: 0.6916
Epoch 25/30
129/129 [==============================] - 44s 342ms/step - loss: 0.4726 - accuracy: 0.8284 - val_loss: 0.9572 - val_accuracy: 0.7056
Epoch 26/30
129/129 [==============================] - 41s 318ms/step - loss: 0.4762 - accuracy: 0.8360 - val_loss: 0.8506 - val_accuracy: 0.7056
Epoch 27/30
129/129 [==============================] - 39s 302ms/step - loss: 0.4734 - accuracy: 0.8216 - val_loss: 1.2935 - val_accuracy: 0.6168
Epoch 28/30
129/129 [==============================] - 39s 300ms/step - loss: 0.4611 - accuracy: 0.8272 - val_loss: 0.8751 - val_accuracy: 0.6869
Epoch 29/30
129/129 [==============================] - 37s 290ms/step - loss: 0.4375 - accuracy: 0.8372 - val_loss: 0.9651 - val_accuracy: 0.6729
Epoch 30/30
129/129 [==============================] - 39s 299ms/step - loss: 0.4292 - accuracy: 0.8501 - val_loss: 1.0778 - val_accuracy: 0.6963
```

Out[41]:

In [42]:
```
cnn.fit(x = training_set , validation_data =test_data , epochs = 30 )
```

```
Epoch 1/30
129/129 [==============================] - 45s 347ms/step - loss: 0.4250 - accuracy: 0.8496 - val_loss: 0.9867 - val_accuracy: 0.6729
Epoch 2/30
129/129 [==============================] - 44s 341ms/step - loss: 0.4170 - accuracy: 0.8469 - val_loss: 1.0115 - val_accuracy: 0.7056
Epoch 3/30
129/129 [==============================] - 44s 341ms/step - loss: 0.4203 - accuracy: 0.8550 - val_loss: 0.8851 - val_accuracy: 0.7150
Epoch 4/30
129/129 [==============================] - 44s 341ms/step - loss: 0.4077 - accuracy: 0.8513 - val_loss: 1.1110 - val_accuracy: 0.6916
Epoch 5/30
129/129 [==============================] - 40s 309ms/step - loss: 0.3930 - accuracy: 0.8603 - val_loss: 1.2546 - val_accuracy: 0.7103
Epoch 6/30
129/129 [==============================] - 42s 327ms/step - loss: 0.4018 - accuracy: 0.8630 - val_loss: 0.9946 - val_accuracy: 0.6916
Epoch 7/30
129/129 [==============================] - 41s 313ms/step - loss: 0.3879 - accuracy: 0.8640 - val_loss: 1.0004 - val_accuracy: 0.7243
Epoch 8/30
129/129 [==============================] - 42s 324ms/step - loss: 0.3729 - accuracy: 0.8655 - val_loss: 1.0725 - val_accuracy: 0.6916
Epoch 9/30
129/129 [==============================] - 41s 319ms/step - loss: 0.3805 - accuracy: 0.8582 - val_loss: 1.0544 - val_accuracy: 0.6916
Epoch 10/30
129/129 [==============================] - 42s 327ms/step - loss: 0.3742 - accuracy: 0.8652 - val_loss: 0.9719 - val_accuracy: 0.6963
Epoch 11/30
129/129 [==============================] - 42s 326ms/step - loss: 0.3737 - accuracy: 0.8686 - val_loss: 0.9270 - val_accuracy: 0.7336
Epoch 12/30
129/129 [==============================] - 43s 334ms/step - loss: 0.3898 - accuracy: 0.8647 - val_loss: 0.9987 - val_accuracy: 0.7196
Epoch 13/30
129/129 [==============================] - 44s 338ms/step - loss: 0.3701 - accuracy: 0.8718 - val_loss: 0.8642 - val_accuracy: 0.7196
Epoch 14/30
129/129 [==============================] - 44s 339ms/step - loss: 0.3546 - accuracy: 0.8786 - val_loss: 1.1820 - val_accuracy: 0.6822
Epoch 15/30
129/129 [==============================] - 50s 390ms/step - loss: 0.3510 - accuracy: 0.8762 - val_loss: 1.0773 - val_accuracy: 0.7150
Epoch 16/30
129/129 [==============================] - 41s 315ms/step - loss: 0.3433 - accuracy: 0.8852 - val_loss: 1.3577 - val_accuracy: 0.7009
Epoch 17/30
129/129 [==============================] - 68s 527ms/step - loss: 0.3400 - accuracy: 0.8796 - val_loss: 1.0770 - val_accuracy: 0.7150
Epoch 18/30
129/129 [==============================] - 63s 477ms/step - loss: 0.3444 - accuracy: 0.8755 - val_loss: 0.9273 - val_accuracy: 0.7243
Epoch 19/30
129/129 [==============================] - 70s 539ms/step - loss: 0.3386 - accuracy: 0.8835 - val_loss: 1.1471 - val_accuracy: 0.6776
Epoch 20/30
129/129 [==============================] - 71s 548ms/step - loss: 0.3300 - accuracy: 0.8869 - val_loss: 1.1275 - val_accuracy: 0.7103
Epoch 21/30
129/129 [==============================] - 77s 599ms/step - loss: 0.3330 - accuracy: 0.8864 - val_loss: 1.2780 - val_accuracy: 0.6963
Epoch 22/30
129/129 [==============================] - 66s 515ms/step - loss: 0.3249 - accuracy: 0.8867 - val_loss: 1.0580 - val_accuracy: 0.7056
Epoch 23/30
129/129 [==============================] - 82s 622ms/step - loss: 0.3225 - accuracy: 0.8903 - val_loss: 1.2799 - val_accuracy: 0.7383
Epoch 24/30
129/129 [==============================] - 101s 785ms/step - loss: 0.3164 - accuracy: 0.8884 - val_loss: 1.3724 - val_accuracy: 0.7056
Epoch 25/30
129/129 [==============================] - 50s 382ms/step - loss: 0.3218 - accuracy: 0.8945 - val_loss: 1.2431 - val_accuracy: 0.7009
Epoch 26/30
129/129 [==============================] - 61s 469ms/step - loss: 0.3212 - accuracy: 0.8945 - val_loss: 0.9750 - val_accuracy: 0.7056
Epoch 27/30
129/129 [==============================] - 111s 851ms/step - loss: 0.3087 - accuracy: 0.9020 - val_loss: 1.4106 - val_accuracy: 0.7056
Epoch 28/30
129/129 [==============================] - 61s 466ms/step - loss: 0.3077 - accuracy: 0.8935 - val_loss: 0.9878 - val_accuracy: 0.7243
Epoch 29/30
129/129 [==============================] - 59s 458ms/step - loss: 0.3071 - accuracy: 0.8976 - val_loss: 1.1608 - val_accuracy: 0.6963
Epoch 30/30
129/129 [==============================] - 38s 295ms/step - loss: 0.3014 - accuracy: 0.8913 - val_loss: 1.4083 - val_accuracy: 0.7336
```

Out[42]:

In [ ]:
```
#preprocess the test image
```

In [43]:
```
import numpy as np
```

In [55]:
```
image = tf.keras.preprocessing.image.load_img("prediction/tu.jpg",target_size=(64,64))
input_arr = tf.keras.preprocessing.image.img_to_array(image)
input_arr = np.expand_dims(input_arr,axis=0)
result = cnn.predict(input_arr)
```

```
1/1 [==============================] - 0s 79ms/step
```

In [52]:
```
training_set.class_indices
```

Out[52]:
```
{'Daisy': 0, 'Dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

In [56]:
```
print(result)
```

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

In [ ]:
```
#Mapping the result to the values
```

In [57]:
```
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("suflower")
elif result[0][4] == 1:
    print("tulip")
```

```
tulip
```

In [ ]:
```

```