```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np
from os import listdir
from os.path import join
import pandas
import cv2
import os
import random
```

In [6]:

```
data_lead = 'D:/IBM Project/Flowers-Dataset/flowers'
folders_lead = os.listdir(data_lead)

print(folders_lead)
```

```
['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
```

In [10]:

```
image_names = []
train_labels = []
train_images = []

size = 64,64

for folder in folders_lead:
    for file in os.listdir(os.path.join(data_lead,folder)):
        if file.endswith("jpg"):
            image_names.append(os.path.join(data_lead,folder,file))
            train_labels.append(folder)
            img = cv2.imread(os.path.join(data_lead,folder,file))
            im = cv2.resize(img,size)
            train_images.append(im)
        else:
            continue
```

In [11]:

```
train = np.array(train_images)

train.shape
```

Out[11]:

```
(4317, 64, 64, 3)
```

In [12]:

```
train = train.astype('float32') / 255.0
```

In [13]:

```
label_dummies = pandas.get_dummies(train_labels)

labels =  label_dummies.values.argmax(1)
```

In [14]:

```
pandas.unique(train_labels)
```

Out[14]:

```
array(['daisy', 'dandelion', 'rose', 'sunflower', 'tulip'], dtype=object)
```

In [15]:

```
union_list = list(zip(train, labels))
random.shuffle(union_list)
train,labels = zip(*union_list)

# Convert the shuffled list to numpy array type

train = np.array(train)
labels = np.array(labels)
```

In [17]:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(64,64,3)),
    keras.layers.Dense(256, activation=tf.nn.relu),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(6, activation=tf.nn.softmax)
])
```

In [18]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 12288)             0

 dense_3 (Dense)             (None, 256)               3145984

 dense_4 (Dense)             (None, 128)               32896

 dense_5 (Dense)             (None, 6)                 774

=================================================================
Total params: 3,179,654
Trainable params: 3,179,654
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
model.compile(optimizer= tf.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [21]:

```
model.fit(train,labels, epochs=15)
```

```
Epoch 1/15
135/135 [==============================] - 4s 33ms/step - loss: 0.9817 -
accuracy: 0.6046
Epoch 2/15
135/135 [==============================] - 4s 33ms/step - loss: 0.9438 -
accuracy: 0.6278
Epoch 3/15
```

```
135/135 [==============================] - 4s 32ms/step - loss: 0.9001 -
accuracy: 0.6416
Epoch 4/15
135/135 [==============================] - 4s 32ms/step - loss: 0.8361 -
accuracy: 0.6854
Epoch 5/15
135/135 [==============================] - 4s 33ms/step - loss: 0.8594 -
accuracy: 0.6674
Epoch 6/15
135/135 [==============================] - 4s 32ms/step - loss: 0.8078 -
accuracy: 0.6787
Epoch 7/15
135/135 [==============================] - 4s 32ms/step - loss: 0.7158 -
accuracy: 0.7239
Epoch 8/15
135/135 [==============================] - 4s 32ms/step - loss: 0.7496 -
accuracy: 0.7130
Epoch 9/15
135/135 [==============================] - 4s 33ms/step - loss: 0.7025 -
accuracy: 0.7308
Epoch 10/15
135/135 [==============================] - 4s 33ms/step - loss: 0.6381 -
accuracy: 0.7640
Epoch 11/15
135/135 [==============================] - 4s 33ms/step - loss: 0.5484 -
accuracy: 0.8024
Epoch 12/15
135/135 [==============================] - 4s 33ms/step - loss: 0.5464 -
accuracy: 0.8024
Epoch 13/15
135/135 [==============================] - 4s 33ms/step - loss: 0.5362 -
accuracy: 0.8110
Epoch 14/15
135/135 [==============================] - 4s 33ms/step - loss: 0.5055 -
accuracy: 0.8114
Epoch 15/15
135/135 [==============================] - 4s 32ms/step - loss: 0.4804 -
accuracy: 0.8279
```

Out[21]:

```
<keras.callbacks.History at 0x1ab1493c670>
```

In [22]:

```
model.save("D:/IBM Project/Flowers-Dataset/flowers.h5")
```

In [ ]: