Importing Dataset

In [2]:

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
ll drive.mount("/content/drive", force_remount=True).
```

Image Augmentation

In [3]:

```
# Importing Library
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [4]:

```
# expanding training and testing variable
train_d=ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=Tr
ue)
test_d=ImageDataGenerator(rescale=1./255)
```

In [5]:

```
#Data augmentation on testing data
vtrain =
train_d.flow_from_directory('/content/drive/MyDrive/flowers/Testing',target
_size=(76,76),class_mode='categorical',batch_size=200)
```

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

In [6]:

```
#Data augmentation on training data
vtest =
test_d.flow_from_directory('/content/drive/MyDrive/flowers/Training',target
_size=(76,76),class_mode='categorical',batch_size=200)
```

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

Creating CNN Model

In [7]:

```
#Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten,
Dense
```

In [8]:

```
#Building a CNN block
model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(76,76,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(500,activation='relu'))
model.add(Dense(250,activation='relu'))
```

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

In [9]:

```
#Compiling the model

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

In [11]:

```
#Fittting the model

model.fit_generator(vtrain,steps_per_epoch=len(vtrain),epochs=15,validation
_data=vtest,validation_steps=len(vtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning
: `Model.fit_generator` is deprecated and will be removed in a future versi
on. Please use `Model.fit`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports
until
Epoch 1/15
22/22 [==============================] - 96s 4s/step - loss: 1.2093 - accur
acy: 0.5038 - val_loss: 1.1528 - val_accuracy: 0.5511
Epoch 2/15
22/22 [==============================] - 69s 3s/step - loss: 1.0804 - accur
acy: 0.5810 - val_loss: 1.0132 - val_accuracy: 0.6115
Epoch 3/15
22/22 [==============================] - 69s 3s/step - loss: 1.0100 - accur
acy: 0.6085 - val_loss: 1.0456 - val_accuracy: 0.6139
Epoch 4/15
22/22 [==============================] - 69s 3s/step - loss: 0.9307 - accur
acy: 0.6467 - val_loss: 0.9421 - val_accuracy: 0.6442
Epoch 5/15
22/22 [==============================] - 70s 3s/step - loss: 0.8715 - accur
acy: 0.6623 - val_loss: 0.9009 - val_accuracy: 0.6674
Epoch 6/15
22/22 [==============================] - 70s 3s/step - loss: 0.8210 - accur
acy: 0.6965 - val_loss: 0.9099 - val_accuracy: 0.6801
Epoch 7/15
22/22 [==============================] - 70s 3s/step - loss: 0.7659 - accur
acy: 0.7169 - val_loss: 0.7732 - val_accuracy: 0.7239
Epoch 8/15
22/22 [==============================] - 72s 3s/step - loss: 0.7307 - accur
acy: 0.7241 - val_loss: 0.7773 - val_accuracy: 0.7037
Epoch 9/15
22/22 [==============================] - 70s 3s/step - loss: 0.7181 - accur
acy: 0.7350 - val_loss: 0.6203 - val_accuracy: 0.7716
Epoch 10/15
22/22 [==============================] - 71s 3s/step - loss: 0.6603 - accur
acy: 0.7528 - val_loss: 0.6906 - val_accuracy: 0.7452
Epoch 11/15
22/22 [==============================] - 70s 3s/step - loss: 0.6256 - accur
acy: 0.7758 - val_loss: 0.7464 - val_accuracy: 0.7253
Epoch 12/15
22/22 [==============================] - 71s 3s/step - loss: 0.5942 - accur
acy: 0.7772 - val_loss: 0.5535 - val_accuracy: 0.8050
Epoch 13/15
22/22 [==============================] - 70s 3s/step - loss: 0.5693 - accur
acy: 0.7957 - val_loss: 0.5050 - val_accuracy: 0.8175
```

```
Epoch 14/15
22/22 [==============================] - 70s 3s/step - loss: 0.5345 - accur
acy: 0.8057 - val_loss: 0.5431 - val_accuracy: 0.8059
Epoch 15/15
22/22 [==============================] - 70s 3s/step - loss: 0.5097 - accur
acy: 0.8131 - val_loss: 0.7343 - val_accuracy: 0.7480
```

Out[11]:

```
<keras.callbacks.History at 0x7fb99243cd90>
```

In [12]:

```
# save model
model.save('flowers.h5')
```

Testing model

In [13]:

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

In [45]:

```
# Testing 1.1(daisy)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/daisy/10993818044_4c
19b86c82.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[45]:

```
'daisy'
```

In [46]:

```
# Testing 1.2(daisy)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/daisy/525780443_bba8
12c26a_m.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[46]:

```
'daisy'
```

In [70]:

```
# Testing 2.1(dandelion)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/dandelion/1195255751
_d58b3d3076.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
```

```
op[prediction]
```

Out[70]:

```
'dandelion'
```

In [73]:

```
# Testing 2.2(dandelion)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/dandelion/1297972485
_33266a18d9.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[73]:

```
'dandelion'
```

In [49]:

```
# Testing 3.1(rose)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/rose/7456887736_54e4
ebac03_n.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[49]:

```
'rose'
```

In [50]:

```
# Testing 3.2(rose)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/rose/33411423082_815
0d9254e_n.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[50]:

```
'tulip'
```

In [51]:

```
# Testing 4.1(sunflower)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/sunflower/7012364067
_5ffc7654c9_m.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[51]:

```
'sunflower'
```

In [52]:

```
# Testing 4.2(sunflower)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/sunflower/2720698862
_486d3ec079_m.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[52]:

```
'sunflower'
```

In [53]:

```
# Testing 5.1(tulip)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/tulip/8892851067_792
42a7362_n.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[53]:

```
'tulip'
```

In [54]:

```
# Testing 5.2(tulip)

img =
image.load_img('/content/drive/MyDrive/flowers/Testing/tulip/5546723510_39a
5a10d3a_n.jpg',target_size=(76,76))
x = image.img_to_array(img)
x = np.expand_dims(x,axis=0)
prediction = np.argmax(model.predict(x))
op = ['daisy','dandelion','rose','sunflower','tulip']
op[prediction]
```

Out[54]:

```
'tulip'
```