In [1]:

```
import os
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import datasets, layers, models
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers import Convolution2D
```

In [2]:

```
train_path = "/content/Flowers-Dataset/flowers/Train"
test_path = "/content/Flowers-Dataset/flowers/Test"
```

Assignment 3

1. Image Augmentation

In [3]:

```
x_train = []
sub_path = train_path + "/daisy"
print(sub_path)
for img in os.listdir(sub_path):
  image_path = sub_path + "/" + img
  img_arr = cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_train.append(img)
```

```
/content/Flowers-Dataset/flowers/Train/daisy
```

In [4]:

```
sub_path = train_path + "/dandelion"
print(sub_path)
for img in os.listdir(sub_path):
  image_path = sub_path + "/" + img
  img_arr = cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_train.append(img)
```

```
/content/Flowers-Dataset/flowers/Train/dandelion
```

In [5]:

```
sub_path = train_path + "/rose"
print(sub_path)
for img in os.listdir(sub_path):
  image_path = sub_path + "/" + img
  img_arr = cv2.imread(image_path)
```

```
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_train.append(img)
```

```
/content/Flowers-Dataset/flowers/Train/rose
```

In [6]:

```
sub_path = train_path + "/sunflower"
print(sub_path)
for img in os.listdir(sub_path):
  image_path = sub_path + "/" + img
  img_arr = cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_train.append(img)
```

```
/content/Flowers-Dataset/flowers/Train/sunflower
```

In [7]:

```
sub_path = train_path + "/tulip"
print(sub_path)
for img in os.listdir(sub_path):
  image_path = sub_path + "/" + img
  img_arr = cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_train.append(img)
```

```
/content/Flowers-Dataset/flowers/Train/tulip
```

In [8]:

```
x_test = []
sub_path=test_path+"/daisy"
for img in os.listdir(sub_path):
  image_path=sub_path+"/"+img
  img_arr=cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_test.append(img)
```

In [9]:

```
sub_path=test_path+"/dandelion"
for img in os.listdir(sub_path):
  image_path=sub_path+"/"+img
  img_arr=cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_test.append(img)
```

In [10]:

```
sub_path=test_path+"/rose"
for img in os.listdir(sub_path):
  image_path=sub_path+"/"+img
  img_arr=cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_test.append(img)
```

In [11]:

```
sub_path=test_path+"/sunflower"
for img in os.listdir(sub_path):
  image_path=sub_path+"/"+img
  img_arr=cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_test.append(img)
```

In [12]:

```
sub_path=test_path+"/tulip"
for img in os.listdir(sub_path):
  image_path=sub_path+"/"+img
  img_arr=cv2.imread(image_path)
  img = cv2.cvtColor(img_arr, cv2.COLOR_BGR2RGB)
  img = cv2.resize(img,(224,224))
  img = img.reshape(224,224,3)
  x_test.append(img)
```

In [13]:

```
train_x = np.array(x_train)
test_x = np.array(x_test)

print(train_x.shape)
print(test_x.shape)
```

```
(3192, 224, 224, 3)
(1125, 224, 224, 3)
```

In [14]:

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

In [15]:

```
training_set = train_datagen.flow_from_directory(train_path,
                                                 target_size = (224, 224),
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(test_path,
                                            target_size = (224, 224),
                                            class_mode = 'categorical')
```

```
Found 3192 images belonging to 5 classes.
Found 1125 images belonging to 5 classes.
```

In [16]:

```
train_y = training_set.classes
test_y = test_set.classes
```

In [17]:

```
training_set.class_indices
```

Out[17]:

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

In [18]:

```
classes = ["daisy","dandelion","rose","sunflower","tulip"]
```

In [19]:

```
train_x=train_x/255.0
test_x=test_x/255.0
```

2. Create Model

In [20]:

```
#Building the CNN
# Initialising the CNN
classifier = Sequential()
```

3. Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output)

In [21]:

```
# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(224, 224, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=96, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=64, activation='relu'))
classifier.add(Dense(units=5, activation='softmax')) # softmax for more than 2
```

4. Compile The Model

In [22]:

```
# Compiling the CNN
classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

In [23]:

```
classifier.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 32)       0
 2D)

 flatten (Flatten)           (None, 93312)             0

 dense (Dense)               (None, 128)               11944064

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 96)                12384
```

```
dropout_1 (Dropout)          (None, 96)                0

dense_2 (Dense)              (None, 64)                6208

dense_3 (Dense)              (None, 5)                 325

=================================================================
Total params: 11,973,125
Trainable params: 11,973,125
Non-trainable params: 0
_________________________________________________________________
```

5. Fit The Model

In [24]:

```
classifier.fit(train_x, train_y, epochs=10, validation_data=(test_x, test_y))
```

```
Epoch 1/10
100/100 [==============================] - 172s 2s/step - loss: 1.4816 - accuracy: 0.3506
- val_loss: 1.4133 - val_accuracy: 0.3422
Epoch 2/10
100/100 [==============================] - 177s 2s/step - loss: 1.2649 - accuracy: 0.4574
- val_loss: 1.1625 - val_accuracy: 0.4791
Epoch 3/10
100/100 [==============================] - 173s 2s/step - loss: 1.1021 - accuracy: 0.5467
- val_loss: 1.1519 - val_accuracy: 0.5538
Epoch 4/10
100/100 [==============================] - 176s 2s/step - loss: 0.8770 - accuracy: 0.6454
- val_loss: 1.1342 - val_accuracy: 0.5733
Epoch 5/10
100/100 [==============================] - 173s 2s/step - loss: 0.6691 - accuracy: 0.7378
- val_loss: 1.1589 - val_accuracy: 0.6142
Epoch 6/10
100/100 [==============================] - 173s 2s/step - loss: 0.4551 - accuracy: 0.8349
- val_loss: 1.5508 - val_accuracy: 0.6027
Epoch 7/10
100/100 [==============================] - 174s 2s/step - loss: 0.3381 - accuracy: 0.8819
- val_loss: 1.7510 - val_accuracy: 0.5867
Epoch 8/10
100/100 [==============================] - 175s 2s/step - loss: 0.2338 - accuracy: 0.9173
- val_loss: 1.7031 - val_accuracy: 0.6151
Epoch 9/10
100/100 [==============================] - 174s 2s/step - loss: 0.1846 - accuracy: 0.9455
- val_loss: 1.9242 - val_accuracy: 0.5858
Epoch 10/10
100/100 [==============================] - 173s 2s/step - loss: 0.1756 - accuracy: 0.9561
- val_loss: 1.7766 - val_accuracy: 0.5929
```

Out[24]:

```
<keras.callbacks.History at 0x7f216b117f50>
```

6. Save The Model

In [25]:

```
classifier.save("model.h5")
```

In [26]:

```
loss, acc = classifier.evaluate(test_x, test_y)
```

```
36/36 [==============================] - 15s 405ms/step - loss: 1.7766 - accuracy: 0.5929
```

7. SUCCESSFULLY PREDICTED DAISY IMAGE FROM TEST IMAGES (Test The Model)

Initialize (Test the model)

In [33]:

```
img = "/content/Flowers-Dataset/flowers/Test/daisy/1150395827_6f94a5c6e4_n.jpg"

test = []
img_arr = cv2.imread(img)

img1 = cv2.resize(img_arr,(224,224))
img1 = img1.reshape(224,224,3)

test.append(img1)
test_img = np.array(test)
test_img = test_img/255

pred = classifier.predict(test_img)
print(classes[np.argmax(pred)])
```

```
daisy
```