Assignment-3

Real-Time Communication System Powered by AI for Specially Abled

Student Name	MOWLIDHARAN A
Student Roll no	621319104034
Maximum Marks	2 Marks

Build CNN Model for Classification Of Flowers

1. Image Augmentation

In [4]:
```
#Image Agumentation
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [5]:
```
train_datagen = ImageDataGenerator(rescale=1./255,zoom_range=0.2,horizontal_flip=True,vertical_flip=True)
```

In [6]:
```
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [9]:
```
xtrain = train_datagen.flow_from_directory("/content/flowers",target_size=(64,64),class_mode='categorical',batch_size=100)
```

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

In [10]:
```
xtest = test_datagen.flow_from_directory("/content/flowers",target_size=(64,64),class_mode='categorical',batch_size=100)
```

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

2. Create Model

```
#Create model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

```
model = Sequential()
```

3. Layers

Convolution

```
#Convolution layer
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

Maxpooling

```
#maxpooling
model.add(MaxPooling2D(pool_size=(2,2)))
```

Flatten

```
#Flatten
model.add(Flatten())
```

Dense Layer

```
#Dense layer
model.add(Dense(300,activation='relu')) #hiddenlayer 1
model.add(Dense(300,activation='relu')) #hiddenlayer 2
model.add(Dense(150,activation='relu')) #hiddenlayer 3
```

Output Layer

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

4,5. Compile & Fit the model

```
#Compile the model
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit(xtrain,steps_per_epoch=len(xtrain),epochs=10,validation_data=xtest,validation_steps=len(xtest))
```

```
Epoch 1/10
44/44 [==============================] - 44s 994ms/step - loss: 1.1533 - accuracy: 0.5198 - val_loss: 1.1798 - val_accuracy: 0.5305
Epoch 2/10
44/44 [==============================] - 43s 986ms/step - loss: 1.0581 - accuracy: 0.5775 - val_loss: 1.1080 - val_accuracy: 0.5497
Epoch 3/10
44/44 [==============================] - 45s 1s/step - loss: 1.0018 - accuracy: 0.6025 - val_loss: 1.0916 - val_accuracy: 0.5661
Epoch 4/10
44/44 [==============================] - 43s 970ms/step - loss: 0.9557 - accuracy: 0.6215 - val_loss: 0.9881 - val_accuracy: 0.6273
Epoch 5/10
44/44 [==============================] - 44s 1s/step - loss: 0.9239 - accuracy: 0.6324 - val_loss: 0.9094 - val_accuracy: 0.6447
Epoch 6/10
44/44 [==============================] - 43s 970ms/step - loss: 0.8764 - accuracy: 0.6630 - val_loss: 0.8566 - val_accuracy: 0.6646
Epoch 7/10
44/44 [==============================] - 46s 1s/step - loss: 0.8414 - accuracy: 0.6711 - val_loss: 0.8138 - val_accuracy: 0.6901
Epoch 8/10
44/44 [==============================] - 44s 1s/step - loss: 0.8224 - accuracy: 0.6829 - val_loss: 0.8595 - val_accuracy: 0.6757
Epoch 9/10
44/44 [==============================] - 42s 982ms/step - loss: 0.8199 - accuracy: 0.6782 - val_loss: 0.7714 - val_accuracy: 0.7012
Epoch 10/10
44/44 [==============================] - 44s 1000ms/step - loss: 0.7729 - accuracy: 0.7005 - val_loss: 0.7357 - val_accuracy: 0.7139
```

6. Save the Model

```
#saving
model.save('Flowers.h5')
```

7. Test the model

```
#Testing the model
import numpy as np
from tensorflow.keras.preprocessing import image
```

```
img = image.load_img("/content/flowers/sunflower/1022552036_67d33d5bd8_n.jpg",target_size=(64,64))
```

```
img
```

```
x = image.img_to_array(img)
x
```

```
array([[[199., 198., 214.],
        [201., 200., 216.],
        [201., 200., 216.],
        ...,
        [187., 188., 208.],
        [186., 187., 207.],
        [184., 185., 205.]],

       [[197., 199., 214.],
        [199., 201., 216.],
        [201., 200., 216.],
```

```
        ...,
        [ 45.,  68.,  50.],
        [ 43.,  39.,  30.],
        [ 55.,  45.,  36.]]], dtype=float32)
```

```
x = np.expand_dims(x,axis=0)
x
```

```
array([[[[199., 198., 214.],
         [201., 200., 216.],
         [201., 200., 216.],
         ...,
         [187., 188., 208.],
         [186., 187., 207.],
         [184., 185., 205.]],

        [[197., 199., 214.],
         [199., 201., 216.],
         [201., 200., 216.],
         ...,
         [186., 187., 207.],
         [185., 186., 206.],
         [184., 185., 205.]],

        [[200., 200., 224.],
         [200., 191., 218.],
         [197., 203., 217.],
         ...,
         [188., 186., 207.],
         [188., 186., 207.],
         [187., 185., 206.]],


        ...,
        [ 45.,  68.,  50.],
        [ 43.,  39.,  30.],
        [ 55.,  45.,  36.]]], dtype=float32)
```

```
x = np.expand_dims(x,axis=0)
x
```

```
array([[[[199., 198., 214.],
         [201., 200., 216.],
         [201., 200., 216.],
         ...,
         [187., 188., 208.],
         [186., 187., 207.],
         [184., 185., 205.]],

        [[197., 199., 214.],
         [199., 201., 216.],
         [201., 200., 216.],
         ...,
         [186., 187., 207.],
         [185., 186., 206.],
         [184., 185., 205.]],

        [[200., 200., 224.],
         [200., 191., 218.],
         [197., 203., 217.],
         ...,
         [188., 186., 207.],
         [188., 186., 207.],
         [187., 185., 206.]],
```

```
...,

[[ 64.,  65.,  47.],
 [106., 120.,  95.],
 [116., 136., 109.],
 ...,
 [109., 130.,  97.],
 [117., 133., 106.],
 [104., 108.,  91.]],

[[ 32.,  40.,  27.],
 [116., 145., 114.],
 [123., 141., 117.],
 ...,
 [ 24.,  54.,  28.],
 [ 92., 121.,  90.],
 [ 54.,  25.,  21.]],

[[129., 125.,  18.],
 [147., 148., 143.],
 [114., 132., 110.],
 ...,
 [ 45.,  68.,  50.],
 [ 43.,  39.,  30.],
 [ 55.,  45.,  36.]]]], dtype=float32)
```

```
model.predict(x)
```

```
1/1 [==============================] - 0s 141ms/step
array([[0., 0., 0., 1., 0.]], dtype=float32)
```

```
model.predict(x)
```

```
1/1 [==============================] - 0s 141ms/step
array([[0., 0., 0., 1., 0.]], dtype=float32)
```

```
xtrain.class_indices
```

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

```
op = ['daisy','dandelion','rose','sunflower','tulip']
pred = np.argmax(model.predict(x))
op[pred]
```

```
1/1 [==============================] - 0s 26ms/step
'sunflower'
```

Split the data into training and testing