Importing Libraries

In [49]:
```
import tensorflow as tf
import numpy as np

from keras.preprocessing.image import ImageDataGenerator
```

Image augmentation

Training set

In [50]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
training_set = train_datagen.flow_from_directory('train_set', target_size=(64,64), bat
```

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

Testing set

In [51]:
```
test_datagen = ImageDataGenerator(rescale=1./255)
test_set = test_datagen.flow_from_directory('test_set', target_size=(64,64), batch_siz
```

```
Found 936 images belonging to 5 classes.
```

Creating a CNN Model

In [52]:
```
cnn = tf.keras.models.Sequential()
```

Adding layers

Convolution

In [53]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation='relu', input_sha
```

Maxpooling

In [54]:
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

Flattening

In [55]:
```
cnn.add(tf.keras.layers.Dropout(0.5))
cnn.add(tf.keras.layers.Flatten())
```

Dense

Hidden layers

In [56]:
```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

Output layer

In [57]:
```
cnn.add(tf.keras.layers.Dense(units=5, activation='softmax'))
```

Compiling the model

In [58]:
```
cnn.compile(optimizer='adam',loss='categorical_crossentropy', metrics=['accuracy'])
```

Fitting the model

In [59]:
```
cnn.fit(x=training_set, validation_data=test_set, epochs=30)
```

```
Epoch 1/30
135/135 [==============================] - 49s 359ms/step - loss: 1.3457 - accuracy:
0.4760 - val_loss: 1.0519 - val_accuracy: 0.5844
Epoch 2/30
135/135 [==============================] - 48s 356ms/step - loss: 1.0621 - accuracy:
0.5738 - val_loss: 0.9151 - val_accuracy: 0.6400
Epoch 3/30
135/135 [==============================] - 49s 361ms/step - loss: 0.9928 - accuracy:
0.6152 - val_loss: 0.9985 - val_accuracy: 0.6421
Epoch 4/30
135/135 [==============================] - 48s 358ms/step - loss: 0.9321 - accuracy:
0.6419 - val_loss: 0.8431 - val_accuracy: 0.6848
Epoch 5/30
135/135 [==============================] - 48s 359ms/step - loss: 0.8602 - accuracy:
0.6692 - val_loss: 0.8235 - val_accuracy: 0.7073
Epoch 6/30
135/135 [==============================] - 54s 398ms/step - loss: 0.8336 - accuracy:
0.6873 - val_loss: 0.7040 - val_accuracy: 0.7468
Epoch 7/30
135/135 [==============================] - 53s 393ms/step - loss: 0.7813 - accuracy:
0.6993 - val_loss: 0.7201 - val_accuracy: 0.7297
Epoch 8/30
135/135 [==============================] - 48s 354ms/step - loss: 0.7664 - accuracy:
0.7088 - val_loss: 0.6730 - val_accuracy: 0.7585
Epoch 9/30
135/135 [==============================] - 47s 345ms/step - loss: 0.7477 - accuracy:
0.7153 - val_loss: 0.7443 - val_accuracy: 0.6955
Epoch 10/30
135/135 [==============================] - 47s 347ms/step - loss: 0.7133 - accuracy:
0.7292 - val_loss: 0.7309 - val_accuracy: 0.7030
Epoch 11/30
135/135 [==============================] - 47s 347ms/step - loss: 0.7018 - accuracy:
0.7371 - val_loss: 0.5655 - val_accuracy: 0.7938
Epoch 12/30
135/135 [==============================] - 48s 353ms/step - loss: 0.6545 - accuracy:
0.7501 - val_loss: 0.6275 - val_accuracy: 0.7692
Epoch 13/30
135/135 [==============================] - 48s 357ms/step - loss: 0.6407 - accuracy:
0.7630 - val_loss: 0.5422 - val_accuracy: 0.7895
Epoch 14/30
135/135 [==============================] - 48s 356ms/step - loss: 0.6285 - accuracy:
0.7677 - val_loss: 0.5220 - val_accuracy: 0.8024
Epoch 15/30
135/135 [==============================] - 48s 357ms/step - loss: 0.5902 - accuracy:
0.7820 - val_loss: 0.4745 - val_accuracy: 0.8344
Epoch 16/30
135/135 [==============================] - 48s 357ms/step - loss: 0.5664 - accuracy:
0.7864 - val_loss: 0.4552 - val_accuracy: 0.8312
Epoch 17/30
135/135 [==============================] - 48s 352ms/step - loss: 0.5890 - accuracy:
0.7799 - val_loss: 0.4519 - val_accuracy: 0.8387
Epoch 18/30
135/135 [==============================] - 48s 356ms/step - loss: 0.5315 - accuracy:
0.8045 - val_loss: 0.3634 - val_accuracy: 0.8771
Epoch 19/30
135/135 [==============================] - 48s 355ms/step - loss: 0.5220 - accuracy:
0.8094 - val_loss: 0.4334 - val_accuracy: 0.8462
Epoch 20/30
135/135 [==============================] - 49s 360ms/step - loss: 0.5012 - accuracy:
0.8177 - val_loss: 0.2974 - val_accuracy: 0.9081
```

```
Epoch 21/30
135/135 [==============================] - 49s 361ms/step - loss: 0.4922 - accuracy:
0.8172 - val_loss: 0.3604 - val_accuracy: 0.8761
Epoch 22/30
135/135 [==============================] - 47s 348ms/step - loss: 0.4854 - accuracy:
0.8249 - val_loss: 0.3735 - val_accuracy: 0.8729
Epoch 23/30
135/135 [==============================] - 47s 349ms/step - loss: 0.4688 - accuracy:
0.8242 - val_loss: 0.3613 - val_accuracy: 0.8697
Epoch 24/30
135/135 [==============================] - 51s 377ms/step - loss: 0.4494 - accuracy:
0.8385 - val_loss: 0.3058 - val_accuracy: 0.8921
Epoch 25/30
135/135 [==============================] - 48s 355ms/step - loss: 0.4352 - accuracy:
0.8392 - val_loss: 0.3078 - val_accuracy: 0.8825
Epoch 26/30
135/135 [==============================] - 49s 361ms/step - loss: 0.4327 - accuracy:
0.8385 - val_loss: 0.2417 - val_accuracy: 0.9284
Epoch 27/30
135/135 [==============================] - 47s 349ms/step - loss: 0.3860 - accuracy:
0.8580 - val_loss: 0.2624 - val_accuracy: 0.9113
Epoch 28/30
135/135 [==============================] - 44s 323ms/step - loss: 0.3852 - accuracy:
0.8603 - val_loss: 0.3074 - val_accuracy: 0.8974
Epoch 29/30
135/135 [==============================] - 41s 307ms/step - loss: 0.3863 - accuracy:
0.8603 - val_loss: 0.2073 - val_accuracy: 0.9188
Epoch 30/30
135/135 [==============================] - 45s 335ms/step - loss: 0.3855 - accuracy:
0.8575 - val_loss: 0.2288 - val_accuracy: 0.9274
```

Out[59]:
```
<keras.callbacks.History at 0x1ea896885e0>
```

Saving the model

In [60]:
```
cnn.save("flower_model")
```

```
INFO:tensorflow:Assets written to: flower_model\assets
```

Testing the model

In [61]:
```
from keras.preprocessing import image
test_image = image.load_img('new_inputs/tulip.jfif', target_size=(64,64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis=0)
result = cnn.predict(test_image)
```

In [62]:
```
training_set.class_indices
```

Out[62]:
```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

In [63]:
```
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('Sunflower')
elif result[0][4]==1:
    print('Tulip')
```

```
Tulip
```