In [1]:
```
import numpy as np
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
```

In [2]:
```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
training_set = train_datagen.flow_from_directory(
        'training_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')
```

```
Found 1028 images belonging to 8 classes.
```

In [3]:
```
test_datagen = ImageDataGenerator(rescale=1./255)
test_set = test_datagen.flow_from_directory(
        'test_set',
        target_size=(64, 64),
        batch_size=32,
        class_mode='categorical')
```

```
Found 394 images belonging to 8 classes.
```

In [4]:
```
cnn = tf.keras.models.Sequential()
```

In [5]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64 , kernel_size=3 , activation='relu' , in
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

In [6]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64 , kernel_size=3 , activation='relu' ))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2 , strides=2))
```

In [7]:
```
cnn.add(tf.keras.layers.Dropout(0.5))
```

In [8]:
```
cnn.add(tf.keras.layers.Flatten())
```

In [9]:
```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

In [10]:
```
cnn.add(tf.keras.layers.Dense(units=8 , activation='softmax'))
```

In [11]:
```
cnn.compile(optimizer = 'rmsprop' , loss = 'categorical_crossentropy' , metrics =
```

In [12]:
```
cnn.fit(x = training_set , validation_data = test_set , epochs = 30)
```

```
Epoch 1/30
33/33 [==============================] - 168s 5s/step - loss: 1.8029 - accuracy:
0.3677 - val_loss: 1.2824 - val_accuracy: 0.4670
Epoch 2/30
33/33 [==============================] - 144s 4s/step - loss: 1.2117 - accuracy:
0.5846 - val_loss: 0.8779 - val_accuracy: 0.6701
Epoch 3/30
33/33 [==============================] - 132s 4s/step - loss: 0.9449 - accuracy:
0.6887 - val_loss: 0.6170 - val_accuracy: 0.7893
Epoch 4/30
33/33 [==============================] - 144s 4s/step - loss: 0.7436 - accuracy:
0.7471 - val_loss: 0.3967 - val_accuracy: 0.8756
Epoch 5/30
33/33 [==============================] - 148s 5s/step - loss: 0.6674 - accuracy:
0.7821 - val_loss: 0.4915 - val_accuracy: 0.8223
Epoch 6/30
33/33 [==============================] - 147s 4s/step - loss: 0.5833 - accuracy:
0.8045 - val_loss: 0.2742 - val_accuracy: 0.9061
Epoch 7/30
33/33 [==============================] - 121s 4s/step - loss: 0.4994 - accuracy:
0.8259 - val_loss: 0.2895 - val_accuracy: 0.9112
Epoch 8/30
33/33 [==============================] - 136s 4s/step - loss: 0.4838 - accuracy:
0.8453 - val_loss: 0.2649 - val_accuracy: 0.9112
Epoch 9/30
33/33 [==============================] - 135s 4s/step - loss: 0.4354 - accuracy:
0.8541 - val_loss: 0.3494 - val_accuracy: 0.8909
Epoch 10/30
33/33 [==============================] - 149s 5s/step - loss: 0.3568 - accuracy:
0.8755 - val_loss: 0.2119 - val_accuracy: 0.9213
Epoch 11/30
33/33 [==============================] - 145s 4s/step - loss: 0.3468 - accuracy:
0.8842 - val_loss: 0.7019 - val_accuracy: 0.7614
Epoch 12/30
33/33 [==============================] - 139s 4s/step - loss: 0.3202 - accuracy:
0.8862 - val_loss: 0.1615 - val_accuracy: 0.9543
Epoch 13/30
33/33 [==============================] - 125s 4s/step - loss: 0.3409 - accuracy:
0.8833 - val_loss: 0.1003 - val_accuracy: 0.9645
Epoch 14/30
33/33 [==============================] - 144s 4s/step - loss: 0.2960 - accuracy:
0.8901 - val_loss: 0.1006 - val_accuracy: 0.9772
Epoch 15/30
33/33 [==============================] - 149s 5s/step - loss: 0.2414 - accuracy:
0.9202 - val_loss: 0.1135 - val_accuracy: 0.9670
Epoch 16/30
33/33 [==============================] - 141s 4s/step - loss: 0.2435 - accuracy:
0.9144 - val_loss: 0.1164 - val_accuracy: 0.9670
Epoch 17/30
33/33 [==============================] - 144s 4s/step - loss: 0.2341 - accuracy:
0.9212 - val_loss: 0.1013 - val_accuracy: 0.9695
Epoch 18/30
33/33 [==============================] - 129s 4s/step - loss: 0.1870 - accuracy:
0.9329 - val_loss: 0.1845 - val_accuracy: 0.9340
Epoch 19/30
33/33 [==============================] - 141s 4s/step - loss: 0.1831 - accuracy:
0.9377 - val_loss: 0.1308 - val_accuracy: 0.9391
Epoch 20/30
33/33 [==============================] - 136s 4s/step - loss: 0.1881 - accuracy:
0.9319 - val_loss: 0.0385 - val_accuracy: 0.9898
Epoch 21/30
33/33 [==============================] - 130s 4s/step - loss: 0.2194 - accuracy:
0.9358 - val_loss: 0.0686 - val_accuracy: 0.9797
Epoch 22/30
```

```
33/33 [==============================] - 138s 4s/step - loss: 0.1719 - accuracy:
0.9368 - val_loss: 0.1827 - val_accuracy: 0.9213
Epoch 23/30
33/33 [==============================] - 152s 5s/step - loss: 0.2468 - accuracy:
0.9202 - val_loss: 0.0425 - val_accuracy: 0.9924
Epoch 24/30
33/33 [==============================] - 179s 5s/step - loss: 0.1418 - accuracy:
0.9553 - val_loss: 0.0688 - val_accuracy: 0.9822
Epoch 25/30
33/33 [==============================] - 173s 5s/step - loss: 0.1490 - accuracy:
0.9572 - val_loss: 0.0657 - val_accuracy: 0.9695
Epoch 26/30
33/33 [==============================] - 183s 6s/step - loss: 0.1445 - accuracy:
0.9484 - val_loss: 0.0744 - val_accuracy: 0.9772
Epoch 27/30
33/33 [==============================] - 177s 5s/step - loss: 0.1560 - accuracy:
0.9475 - val_loss: 0.0597 - val_accuracy: 0.9797
Epoch 28/30
33/33 [==============================] - 174s 5s/step - loss: 0.1609 - accuracy:
0.9446 - val_loss: 0.0417 - val_accuracy: 0.9873
Epoch 29/30
33/33 [==============================] - 166s 5s/step - loss: 0.1163 - accuracy:
0.9621 - val_loss: 0.0614 - val_accuracy: 0.9797
Epoch 30/30
33/33 [==============================] - 160s 5s/step - loss: 0.1197 - accuracy:
0.9621 - val_loss: 0.0267 - val_accuracy: 0.9949
```

Out[12]:
```
<keras.callbacks.History at 0x24839e4c220>
```

In [13]:
```
cnn.save("leafindt.h5")
```

In [14]:
```
from tensorflow import keras
model = keras.models.load_model('leafindt.h5')
```

In [20]:
```
from keras_preprocessing import image
import numpy as np

prediction_img= image.load_img('Predection/11.jpg', target_size=(64,64))
prediction_img = image.img_to_array(prediction_img)
prediction_img = np.expand_dims(prediction_img,axis=0)

# predicting image
result = model.predict(prediction_img)
```

```
1/1 [==============================] - 2s 2s/step
```

In [21]:
```
print(result)
```

```
[[0. 0. 0. 0. 0. 0. 1. 0.]]
```

In [23]:
```
if result[0][0]==1:
    print('PAN')
elif result[0][1]==1:
    print('GUAVA')
elif result[0][2]==1:
    print('JOBA')
elif result[0][3]==1:
    print('LEMON')
elif result[0][4]==1:
    print('MANGO')
elif result[0][5]==1:
    print('MINT')
elif result[0][6]==1:
    print('NEEM')
```

```
elif result[0][7]==1:
    print('TULSI')
```

```
NEEM
```