DL Assignment : 2

Name : Shrikrushna Zirape

Roll No : 41283 (BE-2)

Problem Statement : CNN

use any dataset of plant disease and design a plant disease detection system using CNN

In [6]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D
import numpy as np
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import cv2
```

In [7]:
```
IMG_HEIGHT = 64
IMG_WIDTH = 64
DEPTH = 3

train_datagen = ImageDataGenerator(rescale = 1./255,
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True,
    fill_mode="nearest")
```

In [8]:
```
training_set = train_datagen.flow_from_directory('data/cotton/Cotton Disease/train',
                                                 color_mode = "rgb",
                                                 shuffle = True,
                                                 seed = 42,
                                                 target_size=  (IMG_HEIGHT, IMG_WIDTH),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')

# Preprocessing the val set
val_datagen = ImageDataGenerator(rescale = 1./255)
val_set = val_datagen.flow_from_directory('data/cotton/Cotton Disease/val',
                                          color_mode = "rgb",
                                          shuffle = True,
                                          seed =42,
                                          target_size = (IMG_HEIGHT, IMG_WIDTH),
                                          batch_size = 32,
                                          class_mode = 'categorical')
```

```
Found 1951 images belonging to 4 classes.
Found 253 images belonging to 4 classes.
```

In [9]:
```
# Initialising the CNN
cnn = tf.keras.models.Sequential()

# Step 1 - Convolution
cnn.add(tf.keras.layers.Conv2D(filters=32,padding="same",kernel_size=3, activation='relu', input_shape=[64, 64, 3]))

# Step 2 - Pooling
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Adding a second convolutional layer
cnn.add(tf.keras.layers.Conv2D(filters=32,padding='same',kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Adding a second convolutional layer
cnn.add(tf.keras.layers.Conv2D(filters=32,padding='same',kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Step 3 - Flattening
cnn.add(tf.keras.layers.Flatten())

# Step 4 - Full Connection
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))

# Step 5 - Output Layer
cnn.add(tf.keras.layers.Dense(units=4, activation='softmax'))
```

In [10]:
```
cnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 64, 64, 32)        896

 max_pooling2d_4 (MaxPooling  (None, 32, 32, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_5 (MaxPooling  (None, 16, 16, 32)       0
 2D)

 conv2d_6 (Conv2D)           (None, 16, 16, 32)        9248

 max_pooling2d_6 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 128)               262272

 dense_3 (Dense)             (None, 4)                 516

=================================================================
Total params: 282,180
Trainable params: 282,180
Non-trainable params: 0
_________________________________________________________________
```

In [11]:
```
# Compiling the CNN
cnn.compile(optimizer = 'adam',loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

In [13]:
```
# Training the CNN on the Training set and evaluating it on the Test set
```

In [12]:
```
m = cnn.fit_generator(training_set, validation_data = val_set, epochs = 30)
```

```
C:\Users\LENOVO\AppData\Local\Temp/ipykernel_13880/268919757.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
  m = cnn.fit_generator(training_set, validation_data = val_set, epochs = 30)
```

```
Epoch 1/30
61/61 [==============================] - 86s 1s/step - loss: 1.2405 - accuracy: 0.4470 - val_loss: 1.0253 - val_accuracy: 0.5415
Epoch 2/30
61/61 [==============================] - 26s 432ms/step - loss: 0.9975 - accuracy: 0.5864 - val_loss: 0.7960 - val_accuracy: 0.6719
Epoch 3/30
61/61 [==============================] - 28s 460ms/step - loss: 0.8708 - accuracy: 0.6402 - val_loss: 0.9708 - val_accuracy: 0.6206
Epoch 4/30
61/61 [==============================] - 27s 444ms/step - loss: 0.7306 - accuracy: 0.7022 - val_loss: 0.5799 - val_accuracy: 0.7668
Epoch 5/30
61/61 [==============================] - 28s 454ms/step - loss: 0.6470 - accuracy: 0.7350 - val_loss: 0.4603 - val_accuracy: 0.8182
Epoch 6/30
61/61 [==============================] - 25s 409ms/step - loss: 0.5917 - accuracy: 0.7632 - val_loss: 0.4643 - val_accuracy: 0.7984
Epoch 7/30
61/61 [==============================] - 24s 390ms/step - loss: 0.5461 - accuracy: 0.7832 - val_loss: 0.4843 - val_accuracy: 0.7984
Epoch 8/30
61/61 [==============================] - 24s 395ms/step - loss: 0.4865 - accuracy: 0.8119 - val_loss: 0.3599 - val_accuracy: 0.8498
Epoch 9/30
61/61 [==============================] - 24s 385ms/step - loss: 0.4642 - accuracy: 0.8191 - val_loss: 0.4955 - val_accuracy: 0.7905
Epoch 10/30
61/61 [==============================] - 24s 386ms/step - loss: 0.4373 - accuracy: 0.8329 - val_loss: 0.3485 - val_accuracy: 0.8656
Epoch 11/30
61/61 [==============================] - 23s 383ms/step - loss: 0.4401 - accuracy: 0.8303 - val_loss: 0.3005 - val_accuracy: 0.8656
Epoch 12/30
61/61 [==============================] - 24s 385ms/step - loss: 0.3968 - accuracy: 0.8385 - val_loss: 0.3458 - val_accuracy: 0.8656
Epoch 13/30
61/61 [==============================] - 24s 390ms/step - loss: 0.4515 - accuracy: 0.8242 - val_loss: 0.3542 - val_accuracy: 0.8696
Epoch 14/30
61/61 [==============================] - 23s 383ms/step - loss: 0.3867 - accuracy: 0.8426 - val_loss: 0.4662 - val_accuracy: 0.7945
Epoch 15/30
61/61 [==============================] - 23s 383ms/step - loss: 0.3914 - accuracy: 0.8411 - val_loss: 0.2889 - val_accuracy: 0.8656
Epoch 16/30
61/61 [==============================] - 24s 394ms/step - loss: 0.3469 - accuracy: 0.8637 - val_loss: 0.2448 - val_accuracy: 0.9012
Epoch 17/30
61/61 [==============================] - 25s 404ms/step - loss: 0.3816 - accuracy: 0.8416 - val_loss: 0.2451 - val_accuracy: 0.8972
Epoch 18/30
61/61 [==============================] - 25s 402ms/step - loss: 0.3133 - accuracy: 0.8862 - val_loss: 0.2318 - val_accuracy: 0.9209
Epoch 19/30
61/61 [==============================] - 25s 413ms/step - loss: 0.3491 - accuracy: 0.8657 - val_loss: 0.2413 - val_accuracy: 0.8893
Epoch 20/30
61/61 [==============================] - 47s 774ms/step - loss: 0.3294 - accuracy: 0.8801 - val_loss: 0.2396 - val_accuracy: 0.9051
Epoch 21/30
61/61 [==============================] - 33s 542ms/step - loss: 0.3208 - accuracy: 0.8760 - val_loss: 0.2379 - val_accuracy: 0.9209
Epoch 22/30
61/61 [==============================] - 23s 372ms/step - loss: 0.3203 - accuracy: 0.8857 - val_loss: 0.2446 - val_accuracy: 0.8933
Epoch 23/30
61/61 [==============================] - 23s 381ms/step - loss: 0.2943 - accuracy: 0.8852 - val_loss: 0.2914 - val_accuracy: 0.8972
Epoch 24/30
61/61 [==============================] - 23s 377ms/step - loss: 0.2936 - accuracy: 0.8847 - val_loss: 0.3222 - val_accuracy: 0.8735
Epoch 25/30
61/61 [==============================] - 25s 406ms/step - loss: 0.3038 - accuracy: 0.8857 - val_loss: 0.1978 - val_accuracy: 0.9289
Epoch 26/30
61/61 [==============================] - 25s 401ms/step - loss: 0.2615 - accuracy: 0.8990 - val_loss: 0.2381 - val_accuracy: 0.9091
Epoch 27/30
61/61 [==============================] - 24s 401ms/step - loss: 0.2625 - accuracy: 0.9036 - val_loss: 0.1818 - val_accuracy: 0.9447
Epoch 28/30
61/61 [==============================] - 24s 396ms/step - loss: 0.2718 - accuracy: 0.8995 - val_loss: 0.1964 - val_accuracy: 0.9289
Epoch 29/30
61/61 [==============================] - 24s 397ms/step - loss: 0.2569 - accuracy: 0.9067 - val_loss: 0.2598 - val_accuracy: 0.8893
Epoch 30/30
61/61 [==============================] - 24s 397ms/step - loss: 0.2582 - accuracy: 0.8975 - val_loss: 0.1930 - val_accuracy: 0.9407
```

In [14]:

```
# Preprocessing the Test set
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = val_datagen.flow_from_directory('data/cotton/Cotton Disease/test',
                                           color_mode = "rgb",
                                           shuffle = True,
                                           seed =42,
                                           target_size = (IMG_HEIGHT, IMG_WIDTH),
                                           #batch_size = 32,
                                           class_mode = 'categorical')
```

```
Found 106 images belonging to 4 classes.
```

In [15]:

```
print("[INFO] Calculating model accuracy")
scores = cnn.evaluate(test_set)
print(f"Test Accuracy: {scores[1]}")
```

```
[INFO] Calculating model accuracy
4/4 [==============================] - 7s 2s/step - loss: 0.1823 - accuracy: 0.9434
Test Accuracy: 0.9433962106704712
```