In [1]:

```
import matplotlib.pyplot as plt
import cv2
import numpy as np
import os
import random
from tqdm import tqdm
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling

DATADIR= "D:/tomatoes"

Categories= ["ripe tomato","rotten tomato"]

for category in Categories:
    path = os.path.join(DATADIR,category)  # create path to raw,ripe and rotten
    for img in os.listdir(path):  # iterate over each image per
        img_array = cv2.imread(os.path.join(path,img))
        #cv2.imshow('image',img_array)



IMG_SIZE =100

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))


training_data = []

def create_training_data():
    for category in Categories:  # do

        path = os.path.join(DATADIR,category)
        class_num = Categories.index(category)

        for img in tqdm(os.listdir(path)):
            try:
                img_array = cv2.imread(os.path.join(path,img) ,)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])

            except Exception as e:
                pass


create_training_data()

random.shuffle(training_data)

X = []
y = []

for features,label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
```

```
X=X/255



model = Sequential()
model.add(Conv2D(256,(3,3),input_shape =X.shape[1:], padding='same'))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (2,2)))

model.add(Conv2D(256,(3,3),padding = 'same'))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (2,2)))

model.add(Conv2D(256,(3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(64))

model.add(Dense(2))
model.add(Activation("sigmoid"))



model.compile(loss = "sparse_categorical_crossentropy",optimizer = "adam",metrics=['accurac
model.fit(X,y,batch_size = 32,epochs = 7,validation_split = 0.1)


datadir= "D:/tomatoes"

CATEGORIES= ["test ripe","test rotten"]

for Category in CATEGORIES:
    path1 = os.path.join(datadir,Category)  # create path to raw,ripe and rotten
    for imgt in os.listdir(path1):  # iterate over each image per
        img_arrayt = cv2.imread(os.path.join(path1,imgt))



new_array2 = cv2.resize(img_arrayt, (IMG_SIZE, IMG_SIZE))


test_data = []

def create_test_data():
    for Category in CATEGORIES:  # do

        path1 = os.path.join(datadir,Category)
        class_num2 = CATEGORIES.index(Category)

        for imgt in tqdm(os.listdir(path1)):
            try:
                img_arrayt = cv2.imread(os.path.join(path1,imgt) ,)
                new_array2 = cv2.resize(img_arrayt, (IMG_SIZE, IMG_SIZE))
                test_data.append([new_array2, class_num2])

            except Exception as e:
                pass
```

```
create_test_data()

random.shuffle(test_data)

A = []
b = []

for features,label in training_data:
    A.append(features)
    b.append(label)

A = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 3)

val_loss, val_acc = model.evaluate(A,b)  # evaluate the out of sample data with model
print(val_loss)  # test model's loss (error)
print(val_acc)  # test model's accuracy
```

```
100%|██████████| 343/343 [00:01<00:00, 251.18it/s]
100%|██████████| 496/496 [00:03<00:00, 126.52it/s]

839/839 [==============================] - ETA: 42 - ETA: 39 - ETA: 37 - ET
A: 34 - ETA: 32 - ETA: 31 - ETA: 29 - ETA: 27 - ETA: 26 - ETA: 24 - ETA: 22
- ETA: 21 - ETA: 19 - ETA: 18 - ETA: 17 - ETA: 15 - ETA: 14 - ETA: 12 - ETA:
11 - ETA: 9 - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - ETA:  - 41s 49ms/step
0.1827089825070089
0.9094159713945172

100%|██████████| 19/19 [00:00<00:00, 88.00it/s]
100%|██████████| 21/21 [00:00<00:00, 66.38it/s]
```

In [3]:

```
print("The validation loss is : {} % ".format(val_loss*100))  # model's loss (error)
print("The validation accuracy is :  {} % ".format(val_acc*100))
```

```
The validation loss is : 18.27089825070089 %
The validation accuracy is :  90.94159713945173 %
```

In [ ]: