In [13]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
```

In [14]:
```
train_dir = r'Downloads\New Plant Diseases Dataset(Augmented)\train'
val_dir = r'Downloads\New Plant Diseases Dataset(Augmented)\valid'
```

In [15]:
```
img_size = 224
batch_size = 8
```

In [16]:
```
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(train_dir, target_size=(img_size, img_size), batch_size=b
```

```
Found 600 images belonging to 3 classes.
```

In [17]:
```
val_datagen = ImageDataGenerator(rescale=1./255)
val_generator = val_datagen.flow_from_directory(val_dir, target_size=(img_size, img_size), batch_size=batch_s
```

```
Found 600 images belonging to 3 classes.
```

In [18]:
```
list(train_generator.class_indices)
```

Out[18]:
```
['Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy']
```

In [19]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
```

In [20]:

```
model = Sequential()

model.add((Conv2D(32, (3,3), activation='relu', input_shape=(img_size, img_size, 3))))

model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))

model.add((Conv2D(64, (3,3), activation='relu')))

model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))

model.add((Conv2D(64, (3,3), activation='relu')))

model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))

model.add((Conv2D(128, (3,3), activation='relu')))

model.add(BatchNormalization())
model.add((MaxPooling2D(2,2)))

model.add((Flatten()))

model.add((Dense(128, activation='relu')))
model.add((Dropout(0.2)))

model.add((Dense(64, activation='relu')))
model.add((Dense(train_generator.num_classes, activation='softmax')))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 222, 222, 32)      896       
                                                                 
 batch_normalization_4 (Batc  (None, 222, 222, 32)     128       
 hNormalization)                                                 
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 111, 111, 32)     0         
 2D)                                                             
                                                                 
 conv2d_5 (Conv2D)           (None, 109, 109, 64)      18496     
                                                                 
 batch_normalization_5 (Batc  (None, 109, 109, 64)     256       
 hNormalization)                                                 
                                                                 
 max_pooling2d_5 (MaxPooling  (None, 54, 54, 64)       0         
 2D)                                                             
                                                                 
 conv2d_6 (Conv2D)           (None, 52, 52, 64)        36928     
                                                                 
 batch_normalization_6 (Batc  (None, 52, 52, 64)       256       
 hNormalization)                                                 
                                                                 
 max_pooling2d_6 (MaxPooling  (None, 26, 26, 64)       0         
 2D)                                                             
                                                                 
 conv2d_7 (Conv2D)           (None, 24, 24, 128)       73856     
                                                                 
 batch_normalization_7 (Batc  (None, 24, 24, 128)      512       
 hNormalization)                                                 
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 12, 12, 128)      0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 18432)             0         
                                                                 
 dense_3 (Dense)             (None, 128)               2359424   
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_4 (Dense)             (None, 64)                8256      
 dense_5 (Dense)             (None, 3)                 195

=================================================================
Total params: 2,499,203
Trainable params: 2,498,627
Non-trainable params: 576
_________________________________________________________________
```

In [21]:

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

In [23]:

```
model.fit(train_generator, epochs=2, validation_data=val_generator)
```

```
Epoch 1/2
75/75 [==============================] - 56s 748ms/step - loss: 1.2378 - accuracy: 0.8283 - val_loss: 19.19
43 - val_accuracy: 0.3133
Epoch 2/2
75/75 [==============================] - 55s 739ms/step - loss: 1.5203 - accuracy: 0.8517 - val_loss: 7.811
1 - val_accuracy: 0.3433
```

Out[23]:

```
<keras.callbacks.History at 0x167c16fe610>
```

In [24]:

```
loss, accuracy = model.evaluate(val_generator)
print("Loss :",loss)
print("Accuracy (Test Data) :",accuracy*100)
```

```
75/75 [==============================] - 12s 158ms/step - loss: 7.8111 - accuracy: 0.3433
Loss : 7.81109619140625
Accuracy (Test Data) : 34.33333337306976
```

In [25]:

```
import numpy as np

img_path =r'Downloads\New Plant Diseases Dataset(Augmented)\valid\Tomato___Early_blight\5b86ab6a-3823-4886-85

img = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.
```

In [26]:

```
prediction = model.predict(img_array)
class_names=['Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy']
```

```
1/1 [==============================] - 0s 365ms/step
```

In [27]:

```
predicted_class = np.argmax(prediction)
print(prediction)
print(predicted_class)
print('Predicted class:', class_names[predicted_class])
```

```
[[9.999981e-01 1.911742e-06 1.260711e-08]]
0
Predicted class: Tomato___Bacterial_spot
```

In [ ]: