```
#Nile Pallavi Roll NO: 4217 Div:B
# DL Practical NO.3A

from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img,img_to_array
train_dir = r'C:\Users\Pallavi Nile\Downloads\DL\pract_3\train'
val_dir = r'C:\Users\Pallavi Nile\Downloads\DL\pract_3\valid'
img_size = 224
batch_size = 32

#preprocessing
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(train_dir,
target_size=(img_size,img_size),
batch_size=batch_size,
class_mode='categorical')

val_datagen = ImageDataGenerator(rescale=1./255)
val_generator = val_datagen.flow_from_directory(val_dir,
target_size=(img_size,img_size),
batch_size=batch_size,
class_mode='categorical')


print(list(train_generator.class_indices))

#model building
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,Dropout,
BatchNormalization
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.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])

#training of model
model.fit(train_generator, epochs=50, validation_data=val_generator)

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

#model testing
import numpy as np
img_path =r'C:\Users\Pallavi
Nile\Downloads\DL\pract_3\valid\Tomato___Bacterial_spot\0ab54691-ba9f-4c1f-a69b-
ec0501df4401___GCREC_Bact.Sp 3170.jpg'
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.

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

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

Output:

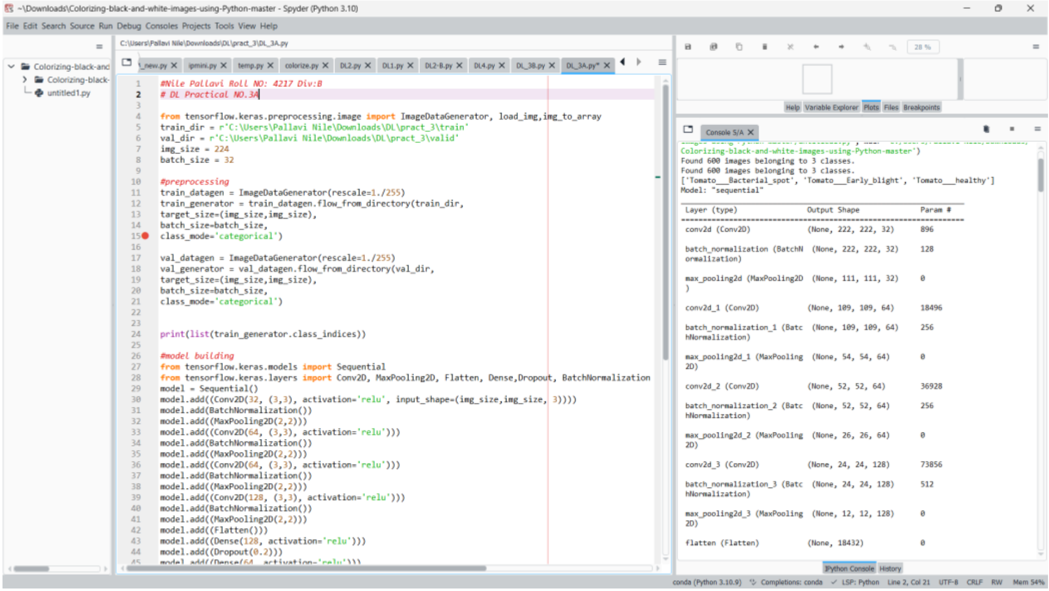
```
Found 600 images belonging to 3 classes.
Found 600 images belonging to 3 classes.
['Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy']
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              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
_________________________________________________________________
Epoch 1/50
19/19 [==============================] - 92s 5s/step - loss: 1.1134 - accuracy: 0.7567 -
val_loss: 1.7457 - val_accuracy: 0.3817
Epoch 2/50
19/19 [==============================] - 120s 6s/step - loss: 0.5225 - accuracy: 0.9117 -
val_loss: 1.0388 - val_accuracy: 0.5467
Epoch 3/50
19/19 [==============================] - 84s 4s/step - loss: 0.3798 - accuracy: 0.9133 -
val_loss: 3.3749 - val_accuracy: 0.3333
Epoch 4/50
19/19 [==============================] - 53s 3s/step - loss: 0.2983 - accuracy: 0.9183 -
val_loss: 10.1802 - val_accuracy: 0.3333
Epoch 5/50
19/19 [==============================] - 60s 3s/step - loss: 0.1306 - accuracy: 0.9583 -
val_loss: 14.1626 - val_accuracy: 0.3333
Epoch 6/50
19/19 [==============================] - 47s 2s/step - loss: 0.1297 - accuracy: 0.9700 -
val_loss: 10.3084 - val_accuracy: 0.4250
```

```
Epoch 7/50
19/19 [==============================] - 37s 2s/step - loss: 0.0608 - accuracy: 0.9783 -
val_loss: 7.7283 - val_accuracy: 0.2700
Epoch 8/50
19/19 [==============================] - 42s 2s/step - loss: 0.1580 - accuracy: 0.9617 -
val_loss: 3.6830 - val_accuracy: 0.4333
Epoch 9/50
19/19 [==============================] - 39s 2s/step - loss: 0.0546 - accuracy: 0.9833 -
val_loss: 4.4959 - val_accuracy: 0.4350
Epoch 10/50
19/19 [==============================] - 40s 2s/step - loss: 0.0226 - accuracy: 0.9933 -
val_loss: 6.3895 - val_accuracy: 0.4767
Epoch 11/50
19/19 [==============================] - 42s 2s/step - loss: 0.0817 - accuracy: 0.9767 -
val_loss: 4.2788 - val_accuracy: 0.4817
Epoch 12/50
19/19 [==============================] - 40s 2s/step - loss: 0.0901 - accuracy: 0.9717 -
val_loss: 6.7955 - val_accuracy: 0.5150
Epoch 13/50
19/19 [==============================] - 43s 2s/step - loss: 0.0667 - accuracy: 0.9800 -
val_loss: 2.0756 - val_accuracy: 0.4967
Epoch 14/50
19/19 [==============================] - 40s 2s/step - loss: 0.1474 - accuracy: 0.9617 -
val_loss: 3.0550 - val_accuracy: 0.5267
Epoch 15/50
19/19 [==============================] - 38s 2s/step - loss: 0.1481 - accuracy: 0.9667 -
val_loss: 12.0756 - val_accuracy: 0.3717
Epoch 16/50
19/19 [==============================] - 37s 2s/step - loss: 0.1517 - accuracy: 0.9717 -
val_loss: 11.6753 - val_accuracy: 0.5100
Epoch 17/50
19/19 [==============================] - 38s 2s/step - loss: 0.1017 - accuracy: 0.9783 -
val_loss: 10.5598 - val_accuracy: 0.3800
Epoch 18/50
19/19 [==============================] - 37s 2s/step - loss: 0.1428 - accuracy: 0.9767 -
val_loss: 5.1567 - val_accuracy: 0.5233
Epoch 19/50
```

```
19/19 [==============================] - 37s 2s/step - loss: 0.0945 - accuracy: 0.9767 -
val_loss: 3.9768 - val_accuracy: 0.5983
Epoch 20/50
19/19 [==============================] - 37s 2s/step - loss: 0.0975 - accuracy: 0.9683 -
val_loss: 8.6274 - val_accuracy: 0.4317
Epoch 21/50
19/19 [==============================] - 36s 2s/step - loss: 0.1087 - accuracy: 0.9767 -
val_loss: 3.5811 - val_accuracy: 0.5867
Epoch 22/50
19/19 [==============================] - 37s 2s/step - loss: 0.1602 - accuracy: 0.9767 -
val_loss: 4.8522 - val_accuracy: 0.6517
Epoch 23/50
19/19 [==============================] - 36s 2s/step - loss: 0.1764 - accuracy: 0.9650 -
val_loss: 1.4186 - val_accuracy: 0.8383
Epoch 24/50
19/19 [==============================] - 37s 2s/step - loss: 0.1202 - accuracy: 0.9783 -
val_loss: 2.0933 - val_accuracy: 0.8733
Epoch 25/50
19/19 [==============================] - 37s 2s/step - loss: 0.0925 - accuracy: 0.9850 -
val_loss: 1.0901 - val_accuracy: 0.8850
Epoch 26/50
19/19 [==============================] - 37s 2s/step - loss: 0.0865 - accuracy: 0.9833 -
val_loss: 1.6068 - val_accuracy: 0.8600
Epoch 27/50
19/19 [==============================] - 44s 2s/step - loss: 0.0886 - accuracy: 0.9817 -
val_loss: 2.7704 - val_accuracy: 0.7800
Epoch 28/50
19/19 [==============================] - 41s 2s/step - loss: 0.2818 - accuracy: 0.9583 -
val_loss: 2.5845 - val_accuracy: 0.8683
Epoch 29/50
19/19 [==============================] - 37s 2s/step - loss: 0.1771 - accuracy: 0.9817 -
val_loss: 0.9145 - val_accuracy: 0.8950
Epoch 30/50
19/19 [==============================] - 44s 2s/step - loss: 0.1389 - accuracy: 0.9850 -
val_loss: 1.4133 - val_accuracy: 0.8517
Epoch 31/50
19/19 [==============================] - 40s 2s/step - loss: 0.0950 - accuracy: 0.9883 -
val_loss: 2.3810 - val_accuracy: 0.7650
```

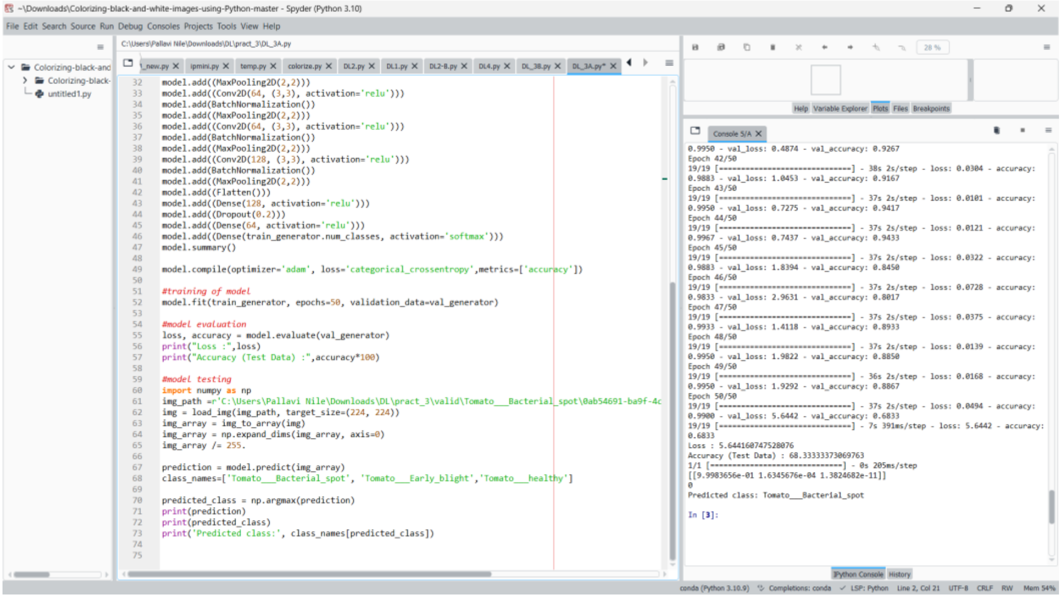
```
Epoch 32/50
19/19 [==============================] - 38s 2s/step - loss: 0.0426 - accuracy: 0.9917 -
val_loss: 1.5762 - val_accuracy: 0.8233
Epoch 33/50
19/19 [==============================] - 43s 2s/step - loss: 0.1423 - accuracy: 0.9850 -
val_loss: 1.4126 - val_accuracy: 0.8950
Epoch 34/50
19/19 [==============================] - 46s 2s/step - loss: 0.0357 - accuracy: 0.9900 -
val_loss: 1.7652 - val_accuracy: 0.8383
Epoch 35/50
19/19 [==============================] - 44s 2s/step - loss: 0.0834 - accuracy: 0.9867 -
val_loss: 1.0388 - val_accuracy: 0.9067
Epoch 36/50
19/19 [==============================] - 40s 2s/step - loss: 0.0499 - accuracy: 0.9867 -
val_loss: 2.2424 - val_accuracy: 0.8350
Epoch 37/50
19/19 [==============================] - 40s 2s/step - loss: 0.0086 - accuracy: 0.9983 -
val_loss: 0.6219 - val_accuracy: 0.9283
Epoch 38/50
19/19 [==============================] - 39s 2s/step - loss: 0.0202 - accuracy: 0.9933 -
val_loss: 0.7721 - val_accuracy: 0.9183
Epoch 39/50
19/19 [==============================] - 40s 2s/step - loss: 0.0281 - accuracy: 0.9950 -
val_loss: 0.4594 - val_accuracy: 0.9300
Epoch 40/50
19/19 [==============================] - 43s 2s/step - loss: 0.0098 - accuracy: 0.9967 -
val_loss: 0.9981 - val_accuracy: 0.9050
Epoch 41/50
19/19 [==============================] - 41s 2s/step - loss: 0.0113 - accuracy: 0.9950 -
val_loss: 0.4874 - val_accuracy: 0.9267
Epoch 42/50
19/19 [==============================] - 38s 2s/step - loss: 0.0304 - accuracy: 0.9883 -
val_loss: 1.0453 - val_accuracy: 0.9167
Epoch 43/50
19/19 [==============================] - 37s 2s/step - loss: 0.0101 - accuracy: 0.9950 -
val_loss: 0.7275 - val_accuracy: 0.9417
Epoch 44/50
```

```
19/19 [==============================] - 37s 2s/step - loss: 0.0121 - accuracy: 0.9967 -
val_loss: 0.7437 - val_accuracy: 0.9433
Epoch 45/50
19/19 [==============================] - 37s 2s/step - loss: 0.0322 - accuracy: 0.9883 -
val_loss: 1.8394 - val_accuracy: 0.8450
Epoch 46/50
19/19 [==============================] - 37s 2s/step - loss: 0.0728 - accuracy: 0.9833 -
val_loss: 2.9631 - val_accuracy: 0.8017
Epoch 47/50
19/19 [==============================] - 37s 2s/step - loss: 0.0375 - accuracy: 0.9933 -
val_loss: 1.4118 - val_accuracy: 0.8933
Epoch 48/50
19/19 [==============================] - 37s 2s/step - loss: 0.0139 - accuracy: 0.9950 -
val_loss: 1.9822 - val_accuracy: 0.8850
Epoch 49/50
19/19 [==============================] - 36s 2s/step - loss: 0.0168 - accuracy: 0.9950 -
val_loss: 1.9292 - val_accuracy: 0.8867
Epoch 50/50
19/19 [==============================] - 37s 2s/step - loss: 0.0494 - accuracy: 0.9900 -
val_loss: 5.6442 - val_accuracy: 0.6833
19/19 [==============================] - 7s 391ms/step - loss: 5.6442 - accuracy: 0.6833
Loss : 5.644160747528076
Accuracy (Test Data) : 68.33333373069763
1/1 [==============================] - 0s 205ms/step
[[9.9983656e-01 1.6345676e-04 1.3824682e-11]]
0
Predicted class: Tomato___Bacterial_spot
```


~\Downloads\Colorizing-black-and-white-images-using-Python-master - Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Pallavi Nile\Downloads\DL\pract_3\DL_3A.py
_new.py ipmini.py temp.py colorize.py DL2.py DL1.py DL2-B.py DL4.py DL_3B.py DL_3A.py*
Colorizing-black-and
Colorizing-black-
untitled1.py
Help Variable Explorer Plots Files Breakpoints
Console 5/A
IPython Console History
conda (Python 3.10.9) Completions: conda LSP: Python Line 2, Col 21 UTF-8 CRLF RW Mem 54%

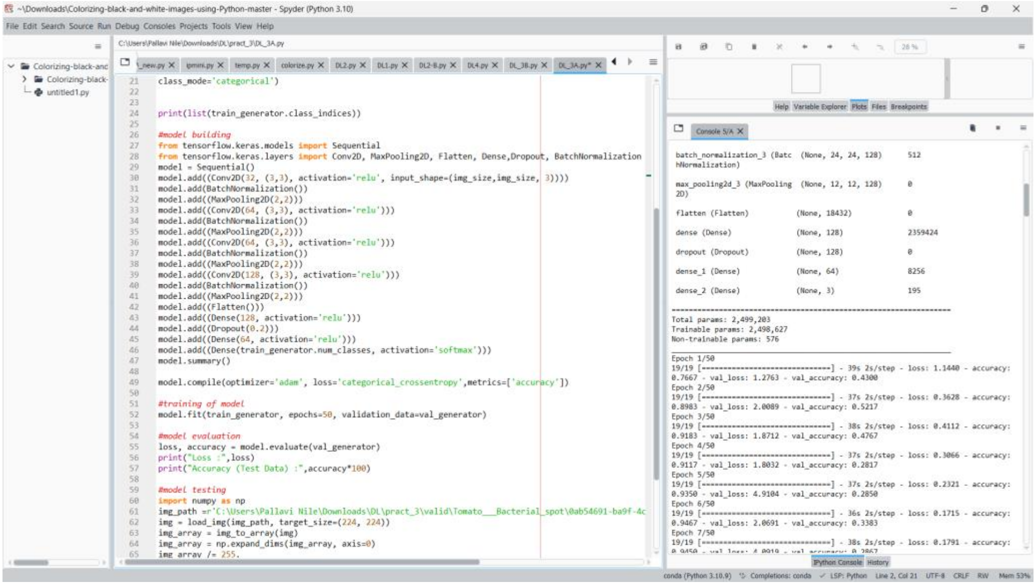
```python
#Nile Pallavi Roll NO: 4217 Div:B
# DL Practical NO.3A

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,img_to_array
train_dir = r'C:\Users\Pallavi Nile\Downloads\DL\pract_3\train'
val_dir = r'C:\Users\Pallavi Nile\Downloads\DL\pract_3\valid'
img_size = 224
batch_size = 32

#preprocessing
train_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(train_dir,
target_size=(img_size,img_size),
batch_size=batch_size,
class_mode='categorical')

val_datagen = ImageDataGenerator(rescale=1./255)
val_generator = val_datagen.flow_from_directory(val_dir,
target_size=(img_size,img_size),
batch_size=batch_size,
class_mode='categorical')


print(list(train_generator.class_indices))

#model building
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,Dropout, BatchNormalization
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')))
```

```
Colorizing-black-and-white-images-using-Python-master')
Found 600 images belonging to 3 classes.
Found 600 images belonging to 3 classes.
['Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___healthy']
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896

 batch_normalization (BatchN  (None, 222, 222, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496

 batch_normalization_1 (Batc  (None, 109, 109, 64)     256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 64)        36928

 batch_normalization_2 (Batc  (None, 52, 52, 64)       256
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 24, 24, 128)       73856

 batch_normalization_3 (Batc  (None, 24, 24, 128)      512
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 128)      0
 2D)

 flatten (Flatten)           (None, 18432)             0
```


~\Downloads\Colorizing-black-and-white-images-using-Python-master - Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Pallavi Nile\Downloads\DL\pract_3\DL_3A.py
_new.py ipmini.py temp.py colorize.py DL2.py DL1.py DL2-B.py DL4.py DL_3B.py DL_3A.py*
Colorizing-black-and
Colorizing-black-
untitled1.py
Help Variable Explorer Plots Files Breakpoints
Console 5/A
IPython Console History
conda (Python 3.10.9) Completions: conda LSP: Python Line 2, Col 21 UTF-8 CRLF RW Mem 54%

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

#training of model
model.fit(train_generator, epochs=50, validation_data=val_generator)

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

#model testing
import numpy as np
img_path =r'C:\Users\Pallavi Nile\Downloads\DL\pract_3\valid\Tomato___Bacterial_spot\0ab54691-ba9f-4c
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.

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

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

```
0.9950 - val_loss: 0.4874 - val_accuracy: 0.9267
Epoch 42/50
19/19 [==============================] - 38s 2s/step - loss: 0.0304 - accuracy:
0.9883 - val_loss: 1.0453 - val_accuracy: 0.9167
Epoch 43/50
19/19 [==============================] - 37s 2s/step - loss: 0.0101 - accuracy:
0.9950 - val_loss: 0.7275 - val_accuracy: 0.9417
Epoch 44/50
19/19 [==============================] - 37s 2s/step - loss: 0.0121 - accuracy:
0.9967 - val_loss: 0.7437 - val_accuracy: 0.9433
Epoch 45/50
19/19 [==============================] - 37s 2s/step - loss: 0.0322 - accuracy:
0.9883 - val_loss: 1.8394 - val_accuracy: 0.8450
Epoch 46/50
19/19 [==============================] - 37s 2s/step - loss: 0.0728 - accuracy:
0.9833 - val_loss: 2.9631 - val_accuracy: 0.8017
Epoch 47/50
19/19 [==============================] - 37s 2s/step - loss: 0.0375 - accuracy:
0.9933 - val_loss: 1.4118 - val_accuracy: 0.8933
Epoch 48/50
19/19 [==============================] - 37s 2s/step - loss: 0.0139 - accuracy:
0.9950 - val_loss: 1.9822 - val_accuracy: 0.8850
Epoch 49/50
19/19 [==============================] - 36s 2s/step - loss: 0.0168 - accuracy:
0.9950 - val_loss: 1.9292 - val_accuracy: 0.8867
Epoch 50/50
19/19 [==============================] - 37s 2s/step - loss: 0.0494 - accuracy:
0.9900 - val_loss: 5.6442 - val_accuracy: 0.6833
19/19 [==============================] - 7s 391ms/step - loss: 5.6442 - accuracy:
0.6833
Loss : 5.644160747528076
Accuracy (Test Data) : 68.33333373069763
1/1 [==============================] - 0s 205ms/step
[[9.9983656e-01 1.6345676e-04 1.3824682e-11]]
0
Predicted class: Tomato___Bacterial_spot

In [3]:
```



```python
    class_mode='categorical')


print(list(train_generator.class_indices))

#model building
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,Dropout, BatchNormalization
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.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])

#training of model
model.fit(train_generator, epochs=50, validation_data=val_generator)

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

#model testing
import numpy as np
img_path =r'C:\Users\Pallavi Nile\Downloads\DL\pract_3\valid\Tomato___Bacterial_spot\0ab54691-ba9f-4c
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.
```

```
 batch_normalization_3 (Batc  (None, 24, 24, 128)      512
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 12, 12, 128)      0
 2D)

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 128)               2359424

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 3)                 195

=================================================================
Total params: 2,499,203
Trainable params: 2,498,627
Non-trainable params: 576
_________________________________________________________________
Epoch 1/50
19/19 [==============================] - 39s 2s/step - loss: 1.1440 - accuracy:
0.7667 - val_loss: 1.2763 - val_accuracy: 0.4300
Epoch 2/50
19/19 [==============================] - 37s 2s/step - loss: 0.3628 - accuracy:
0.8983 - val_loss: 2.0089 - val_accuracy: 0.5217
Epoch 3/50
19/19 [==============================] - 38s 2s/step - loss: 0.4112 - accuracy:
0.9183 - val_loss: 1.8712 - val_accuracy: 0.4767
Epoch 4/50
19/19 [==============================] - 37s 2s/step - loss: 0.3066 - accuracy:
0.9117 - val_loss: 1.8032 - val_accuracy: 0.2817
Epoch 5/50
19/19 [==============================] - 37s 2s/step - loss: 0.2321 - accuracy:
0.9350 - val_loss: 4.9104 - val_accuracy: 0.2850
Epoch 6/50
19/19 [==============================] - 36s 2s/step - loss: 0.1715 - accuracy:
0.9467 - val_loss: 2.0691 - val_accuracy: 0.3383
Epoch 7/50
19/19 [==============================] - 38s 2s/step - loss: 0.1791 - accuracy:
0.9450 - val_loss: 4.0919 - val_accuracy: 0.2867
```