Data Loading

In [68]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import numpy as np
import os
```

In [69]:
```
IMAGE_SIZE = 230
BATCH_SIZE = 32
CHANNELS = 3
EPOCHS =10
```

In [94]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    'rice_leaf_disease_images',
    shuffle = True,
    image_size = (IMAGE_SIZE, IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
```

```
Found 7910 files belonging to 5 classes.
```

In [95]:
```
class_names = dataset.class_names
class_names
```

Out[95]:
```
['Bacterialblight', 'Blast', 'Brownspot', 'Healthy', 'Tungro']
```

In [96]:
```
len(dataset) #248*32=7936
```

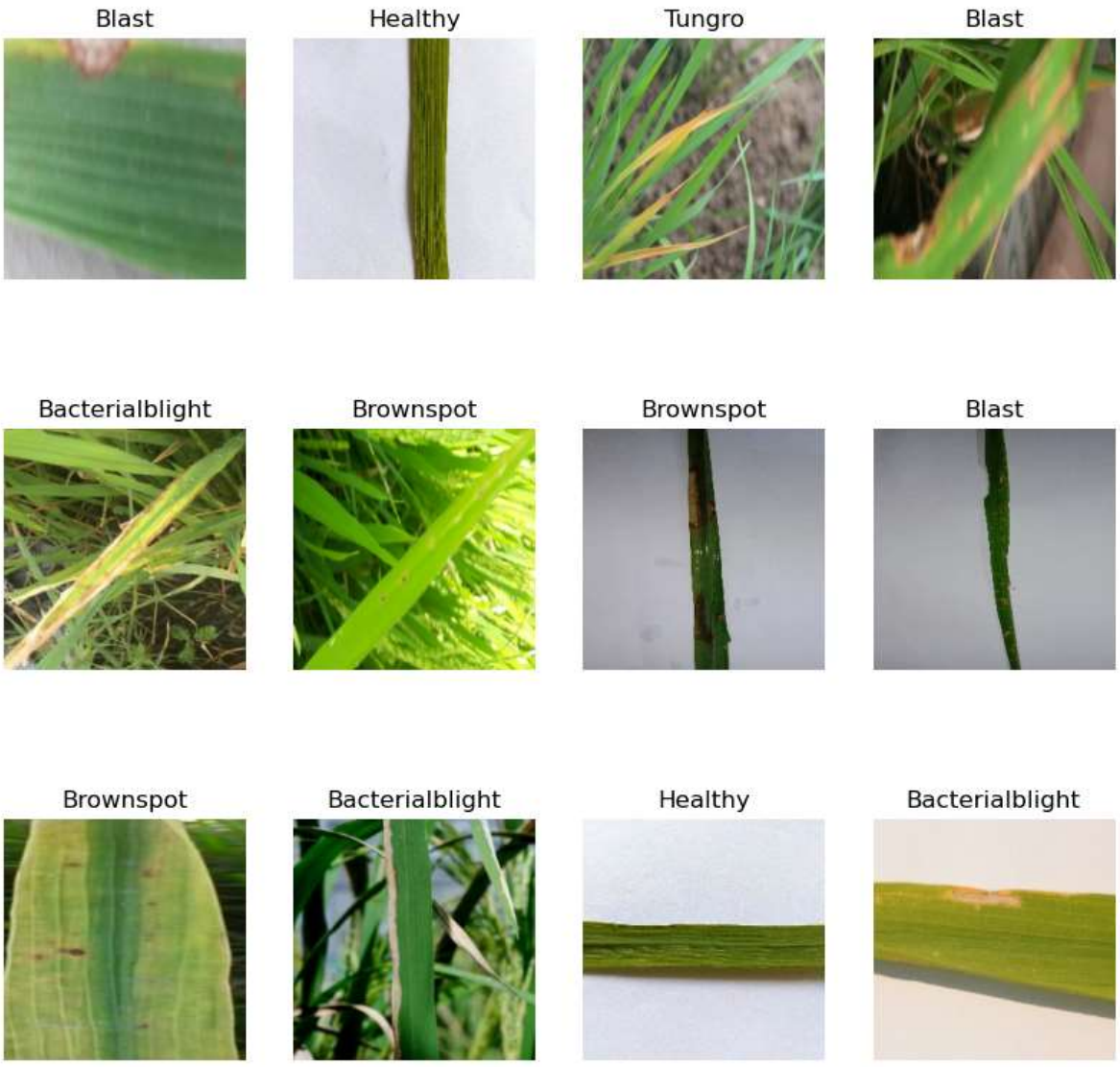
Out[96]:
```
248
```

In [97]:
```
# One random batch of images
for image_batch, label_batch in dataset.take(1):
    print(image_batch.shape)
    print(label_batch.numpy())
```

```
(32, 230, 230, 3)
[3 2 1 0 0 3 1 2 4 3 4 3 4 3 0 0 2 0 2 0 3 0 2 3 2 2 1 2 2 1 2 4]
```

In [98]:
```
plt.figure(figsize=(10,10))
for image_batch, label_batch in dataset.take(1):
    print(image_batch.shape)
    print(label_batch.numpy())
    for i in range(12): #showing 12 images out of 32
        ax = plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[label_batch[i]])
        plt.axis("off")
```

```
(32, 230, 230, 3)
[1 3 4 1 0 2 2 1 2 0 3 0 3 0 3 3 0 0 3 1 2 0 1 0 1 3 2 0 1 2 1 3]
```



In [99]:
```
# (32=batch_size, 256, 256=image_size, 0 to 3=typesofdiseases)
# 0 - Bacterial Blight
# 1 - Blast
# 2 - Brownspot
# 3 - Tungro
```

In [100]:
```
# Spitting dataset for training, validation and testing
# 80% for training 10% for validation and 10% for testing
def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0
    ds_size = len(ds)
    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)
    train_size = int(train_split*ds_size)
    val_size = int(val_split*ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds
```

In [101]:
```
train_ds, val_ds, test_ds =get_dataset_partitions_tf(dataset)
```

In [102]:
```
# Catching and prefeching
train_ds = train_ds.cache().shuffle(1500).prefetch(buffer_size=tf.data.AUTOTUN
val_ds = val_ds.cache().shuffle(1500).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=tf.data.AUTOTUNE)
```

Preprocessing

In [103]:
```
def augment(image, label):
  image = tf.cast(image, tf.float32)
  image = tf.image.resize(image, [IMAGE_SIZE, IMAGE_SIZE])
  image = (image / 255.0)
  image = tf.image.rot90(image)
  image = tf.image.random_brightness(image, max_delta=0.5)
  return image, label


train_ds = (
    train_ds
    .shuffle(1000)
    .map(augment, num_parallel_calls=tf.data.AUTOTUNE)
    .prefetch(tf.data.AUTOTUNE)
)
```

In [104]:
```
# Layer for resizing and rescaling
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

In [105]:
```
# Data Augmentation
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2)
])
```

In [ ]:
```
def show_example(img,label):
    print("label: ",dataset.classes[label],"("+str(label)+")")
    plt.imshow(img.permute(1,2,0))
show_example(*dataset[0])
```

In [ ]:
```
show_example(*dataset[1])
```

ResNet152

In [106]:
```
from tensorflow.keras.applications import ResNet152
```

In [107]:
```
resnet152 = ResNet152(input_shape=(IMAGE_SIZE,IMAGE_SIZE,CHANNELS),weights='im
```

In [108]:
```
# Don't train existing weights
for layer in resnet152.layers:
    layer.trainable = False
```

In [109]:
```
x = tf.keras.layers.Flatten()(resnet152.output)
```

In [110]:
```
prediction = tf.keras.layers.Dense(len(class_names),activation='softmax')(x)
```

In [111]:
```
model = tf.keras.Model(inputs=resnet152.input, outputs=prediction)
```

In [112]:
```
model.summary()
```

```
lock25_1_relu[0][0]']

 conv4_block25_2_bn (BatchNorma  (None, 15, 15, 256)  1024       ['conv4_b
lock25_2_conv[0][0]']
 lization)

 conv4_block25_2_relu (Activati  (None, 15, 15, 256)  0          ['conv4_b
lock25_2_bn[0][0]']
 on)

 conv4_block25_3_conv (Conv2D)  (None, 15, 15, 1024  263168      ['conv4_b
lock25_2_relu[0][0]']
                                )

 conv4_block25_3_bn (BatchNorma  (None, 15, 15, 1024  4096       ['conv4_b
lock25_3_conv[0][0]']
 lization)                      )

 conv4_block25_add (Add)        (None, 15, 15, 1024  0           ['conv4_b
lock24_out[0][0]',
```

In [113]:
```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy'])
```

In [114]:
```
logdir='logs'
```

In [115]:
```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

In [116]:
```
callback = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.001,
    patience=6,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
)
```

In [*]:
```
history = model.fit(
    train_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    callbacks=tensorboard_callback
)
```

```
Epoch 1/10
198/198 [==============================] - 2069s 10s/step - loss: 3.9235 - ac
curacy: 0.5195 - val_loss: 47.3937 - val_accuracy: 0.1979
Epoch 2/10
198/198 [==============================] - 1743s 9s/step - loss: 1.5963 - acc
uracy: 0.6266 - val_loss: 52.6918 - val_accuracy: 0.2018
Epoch 3/10
198/198 [==============================] - 1698s 9s/step - loss: 1.5444 - acc
uracy: 0.6510 - val_loss: 67.9972 - val_accuracy: 0.1992
Epoch 4/10
198/198 [==============================] - 1804s 9s/step - loss: 1.1962 - acc
uracy: 0.7108 - val_loss: 81.9480 - val_accuracy: 0.1992
Epoch 5/10
198/198 [==============================] - 2484s 13s/step - loss: 1.5439 - ac
curacy: 0.6924 - val_loss: 84.4683 - val_accuracy: 0.1979
Epoch 6/10
198/198 [==============================] - 3223s 16s/step - loss: 2.0523 - ac
curacy: 0.6672 - val_loss: 109.5308 - val_accuracy: 0.1992
Epoch 7/10
154/198 [======================>.......] - ETA: 13:05 - loss: 1.2493 - accura
cy: 0.7493
```

In [*]:
```
scores = model.evaluate(test_ds)
scores
```

In [*]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [*]:
```
plt.figure(figsize=(8,8))
plt.subplot(1,2,1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc = 'lower right')
plt.title('Training and Validation Accuracy')
```

In [*]:
```
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    prediction = model.predict(img_array)

    prediction_class = class_names[np.argmax(prediction[0])]
    confidence = round(100 * (np.max(prediction[0])),2)

    return prediction_class, confidence
```

In [*]:
```
plt.figure(figsize=(15,15))
for images, labels in test_ds.take(2):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype('uint8'))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]
        plt.title(f'predicted: {predicted_class},\n confidence: {confidence},
        plt.axis("off")
```

Saving the Trained Model

In [*]:
```
# Saving the model
model_version = max([int(i) for i in os.listdir("new_models") + [0]])+1
model.save(f'new_models\{model_version}')
```

In [*]:
```
# Loading saved model
```

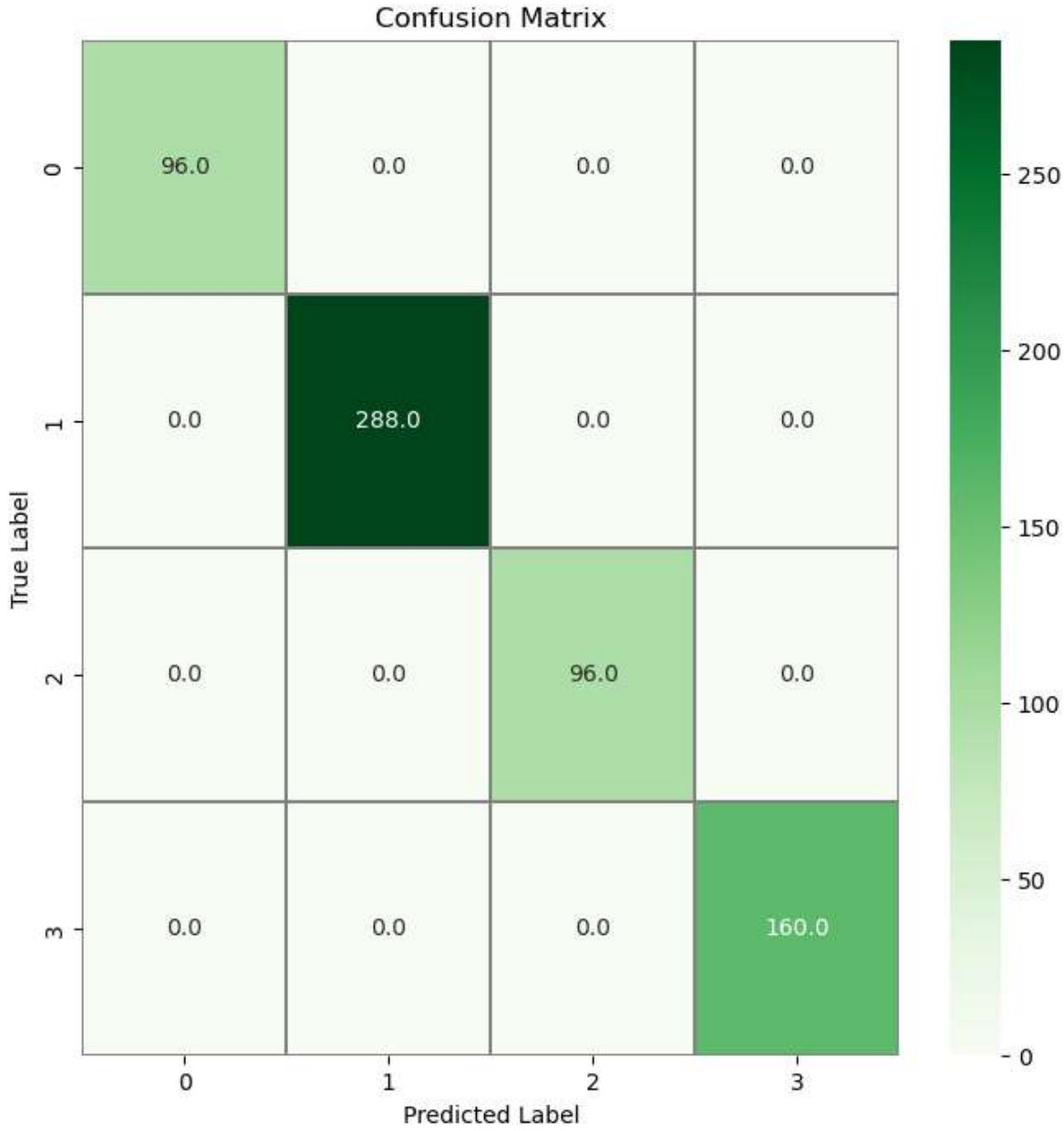
In [ ]:
```
new_model = tf.keras.models.load_model('new_models/22')

# Check its architecture
new_model.summary()
```

Confusion Matrix and Classification Report

In [22]:
```
from sklearn.metrics import confusion_matrix , classification_report
```

In [23]:
```
# confusion matrix
import seaborn as sns
# Predict the values from the validation dataset
Y_pred = new_model.predict(test_ds)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = tf.concat([y for x, y in test_ds], axis=0)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,cmap="Greens",linecolor
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
20/20 [==============================] - 76s 4s/step
```



In [24]:
```
print(classification_report(Y_true, Y_pred_classes, target_names=class_names))
```

```
                precision    recall  f1-score   support

Bacterialblight       1.00      1.00      1.00        96
         Blast        1.00      1.00      1.00       288
     Brownspot        1.00      1.00      1.00        96
        Tungro        1.00      1.00      1.00       160

      accuracy                           1.00       640
     macro avg        1.00      1.00      1.00       640
  weighted avg        1.00      1.00      1.00       640
```

In [ ]:

New Image Classification

In [25]:
```
import numpy as np
import cv2
```

In [66]:
```
img_path = cv2.imread(os.path.join('Testing Images','bb1.jpg'))
img_path = cv2.cvtColor(img_path,cv2.COLOR_BGR2RGB)
plt.imshow(img_path)
```

Out[66]:
```
<matplotlib.image.AxesImage at 0x2229e77c6a0>
```

In [67]:
```
img = cv2.resize(img_path,(224,224))
img = np.reshape(img,[1,224,224,3])
```

In [68]:
```
pred = new_model.predict(img)
```

```
1/1 [==============================] - 0s 370ms/step
```

In [69]:
```
pred
```

Out[69]:
```
array([[1.0000000e+00, 3.7825334e-09, 1.6624246e-24, 8.5504887e-30]],
      dtype=float32)
```

In [70]:
```
prediction_class = class_names[np.argmax(pred)]
prediction_class
```

Out[70]: 'Bacterialblight'

In [ ]:

In [ ]:

In [ ]:

2 - vgg16 2epochs earlystopping 7 - vgg16 8 - vgg16 data + bb.jpg added

In [ ]:

In [ ]: