Data Loading

In [1]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import numpy as np
import os
```

In [2]:
```
IMAGE_SIZE = 224
BATCH_SIZE = 32
CHANNELS = 3
EPOCHS =10
```

In [3]:
```
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 5932 files belonging to 4 classes.
```

In [4]:
```
class_names = dataset.class_names
class_names
```

Out[4]:
```
['Bacterialblight', 'Blast', 'Brownspot', 'Tungro']
```

In [5]:
```
len(dataset) #186*32=5931
```

Out[5]:
```
186
```

In [6]:
```
# One random batch of images
for image_batch, label_batch in dataset.take(1):
    print(image_batch.shape)
    print(label_batch.numpy())
```

```
(32, 224, 224, 3)
[1 2 0 1 2 2 0 0 0 3 1 3 1 2 1 2 3 1 0 1 2 2 2 1 2 0 3 1 0 2 2 3]
```

In [7]:
```
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, 224, 224, 3)
[2 2 0 3 3 1 3 3 2 0 0 0 1 1 1 0 0 3 1 1 2 2 3 1 1 0 2 3 2 0 3 1]
```

In [8]:
```
# (32=batch_size, 256, 256=image_size, 0 to 3=typesofdiseases)
# 0 - Bacterial Blight
# 1 - Blast
# 2 - Brownspot
# 3 - Tungro
```

In [9]:
```
# 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 [10]:
```
train_ds, val_ds, test_ds =get_dataset_partitions_tf(dataset)
```

In [11]:
```
# Catching and prefeching
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUN
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Preprocessing

In [12]:
```
# 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 [13]:
```
# Data Augmentation
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2)
])
```

VGG16

In [14]:
```
from tensorflow.keras.applications.vgg19 import VGG19
```

In [15]:
```
vgg19 = VGG19(input_shape=(IMAGE_SIZE,IMAGE_SIZE,CHANNELS),weights='imagenet',
```

In [16]:
```
# Don't train existing weights
for layer in vgg19.layers:
    layer.trainable = False
```

In [17]:
```
x = tf.keras.layers.Flatten()(vgg19.output)
```

In [18]:
```
prediction = tf.keras.layers.Dense(len(class_names),activation='softmax')(x)
```

In [19]:
```
model = tf.keras.Model(inputs=vgg19.input, outputs=prediction)
```

In [20]:
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv4 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 4)                 100356

=================================================================
Total params: 20,124,740
Trainable params: 100,356
```

```
Non-trainable params: 20,024,384
_________________________________________________________________
```

In [21]:
```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy'])
```

In [22]:
```
logdir='logs'
```

In [23]:
```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

In [24]:
```
history = model.fit(
    train_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    verbose=1,
    validation_data=val_ds,
    callbacks=[tensorboard_callback]
)
```

```
Epoch 1/10
148/148 [==============================] - 1462s 10s/step - loss: 1.4393 - ac
curacy: 0.9092 - val_loss: 0.3233 - val_accuracy: 0.9809
Epoch 2/10
148/148 [==============================] - 1423s 10s/step - loss: 0.2240 - ac
curacy: 0.9862 - val_loss: 0.0071 - val_accuracy: 0.9983
Epoch 3/10
148/148 [==============================] - 1145s 8s/step - loss: 0.0843 - acc
uracy: 0.9922 - val_loss: 4.5133e-06 - val_accuracy: 1.0000
Epoch 4/10
148/148 [==============================] - 1057s 7s/step - loss: 0.0104 - acc
uracy: 0.9985 - val_loss: 3.8078e-07 - val_accuracy: 1.0000
Epoch 5/10
148/148 [==============================] - 1056s 7s/step - loss: 2.9393e-06 -
accuracy: 1.0000 - val_loss: 2.4213e-07 - val_accuracy: 1.0000
Epoch 6/10
148/148 [==============================] - 1068s 7s/step - loss: 6.1962e-07 -
accuracy: 1.0000 - val_loss: 1.9950e-07 - val_accuracy: 1.0000
Epoch 7/10
148/148 [==============================] - 1061s 7s/step - loss: 5.1727e-07 -
accuracy: 1.0000 - val_loss: 1.5335e-07 - val_accuracy: 1.0000
Epoch 8/10
148/148 [==============================] - 1019s 7s/step - loss: 4.3921e-07 -
accuracy: 1.0000 - val_loss: 1.3080e-07 - val_accuracy: 1.0000
Epoch 9/10
148/148 [==============================] - 1022s 7s/step - loss: 3.7806e-07 -
accuracy: 1.0000 - val_loss: 1.1155e-07 - val_accuracy: 1.0000
Epoch 10/10
148/148 [==============================] - 1024s 7s/step - loss: 3.2936e-07 -
accuracy: 1.0000 - val_loss: 9.6856e-08 - val_accuracy: 1.0000
```

In [25]:
```
scores = model.evaluate(test_ds)
scores
```

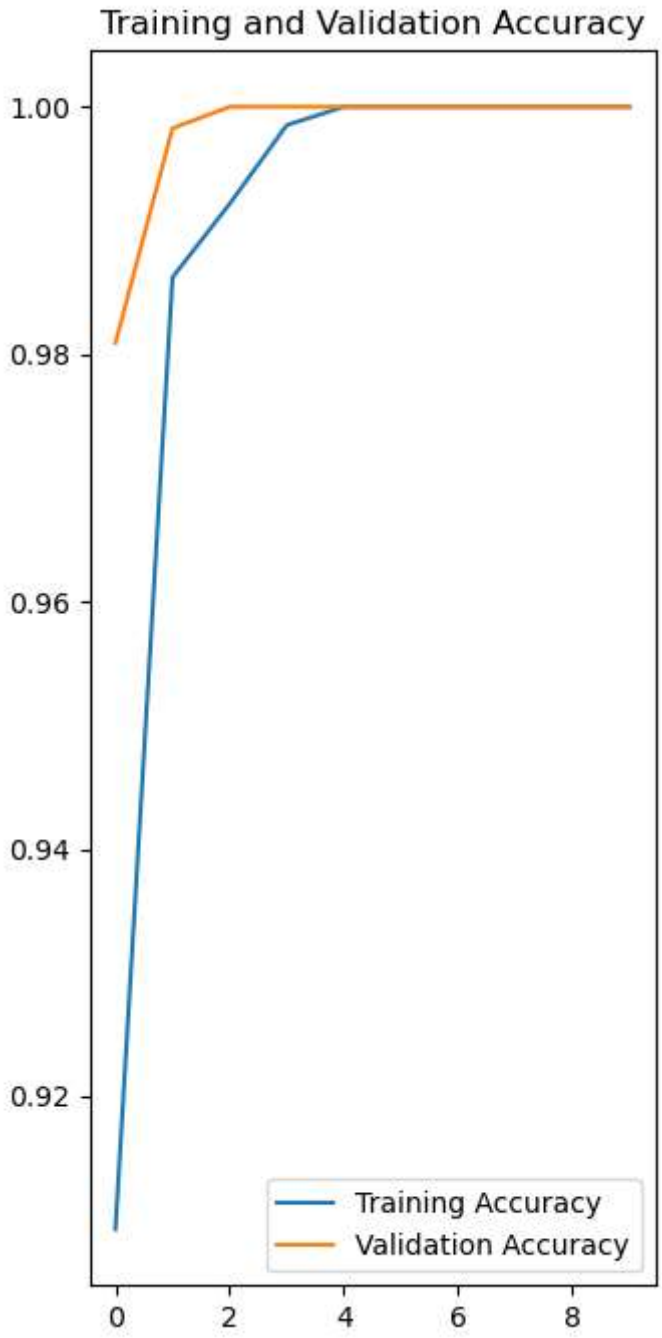
```
20/20 [==============================] - 145s 7s/step - loss: 0.2158 - accura
cy: 0.9969
```

Out[25]:
```
[0.21576246619224548, 0.996874988079071]
```

In [26]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [27]:
```
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')
```

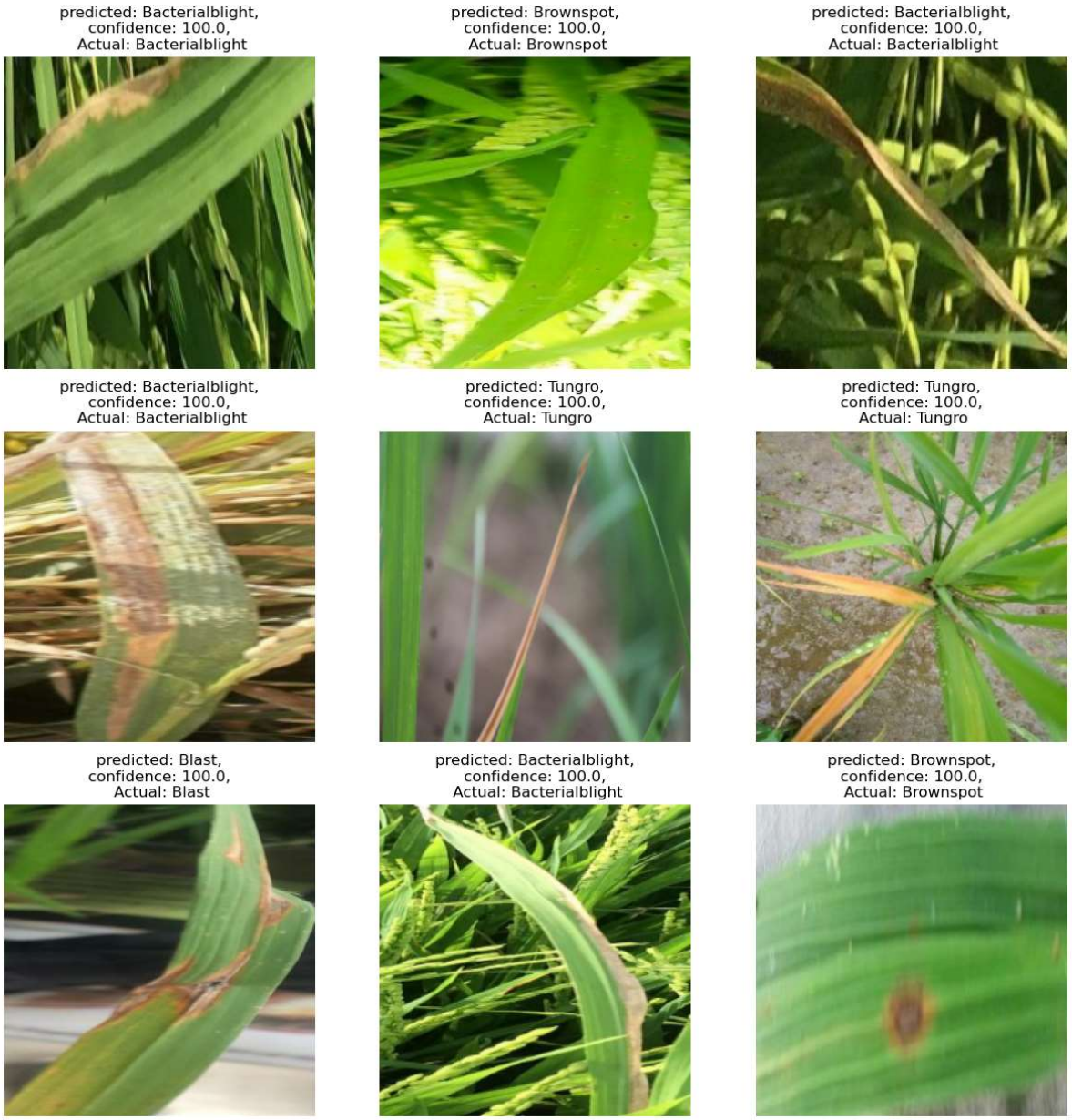
Out[27]:
```
Text(0.5, 1.0, 'Training and Validation Accuracy')
```



In [29]:
```
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 [30]:
```
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")
```

```
1/1 [==============================] - 0s 494ms/step
1/1 [==============================] - 0s 242ms/step
1/1 [==============================] - 0s 257ms/step
1/1 [==============================] - 0s 248ms/step
1/1 [==============================] - 0s 259ms/step
1/1 [==============================] - 0s 261ms/step
1/1 [==============================] - 0s 290ms/step
1/1 [==============================] - 0s 269ms/step
1/1 [==============================] - 0s 266ms/step
1/1 [==============================] - 0s 241ms/step
1/1 [==============================] - 0s 235ms/step
1/1 [==============================] - 0s 235ms/step
1/1 [==============================] - 0s 227ms/step
1/1 [==============================] - 0s 254ms/step
1/1 [==============================] - 0s 232ms/step
1/1 [==============================] - 0s 239ms/step
1/1 [==============================] - 0s 229ms/step
1/1 [==============================] - 0s 233ms/step
```



In [31]:
```
# Saving the model
model_version = max([int(i) for i in os.listdir("models") + [0]])+1
model.save(f'models\{model_version}')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _
jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_conv
olution_op, _jit_compiled_convolution_op while saving (showing 5 of 17). Thes
e functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: models\13\assets

INFO:tensorflow:Assets written to: models\13\assets
```

In [6]:
```
new_model = tf.keras.models.load_model('models/13')

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

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv4 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 4)                 100356

=================================================================
Total params: 20,124,740
Trainable params: 100,356
```

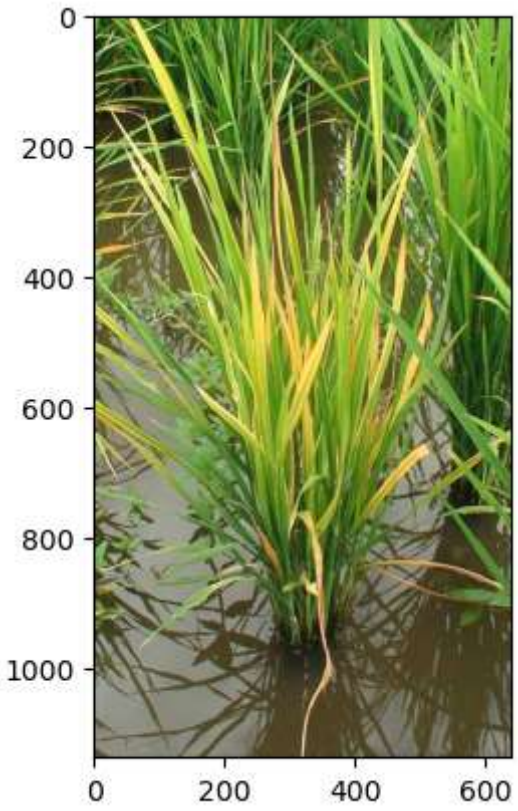
```
Non-trainable params: 20,024,384
_________________________________________________________________
```

In [ ]:

In [7]:
```
import numpy as np
import cv2
```

In [54]:
```
img_path = cv2.imread(os.path.join('Testing Images','t.jfif'))
img_path = cv2.cvtColor(img_path,cv2.COLOR_BGR2RGB)
plt.imshow(img_path)
```

Out[54]:
```
<matplotlib.image.AxesImage at 0x2496df6f970>
```



In [55]:
```
img = cv2.resize(img_path,(224,224))
img = np.reshape(img,[1,224,224,3])
```

In [56]:
```
pred = new_model.predict(img)
```

```
1/1 [==============================] - 0s 217ms/step
```

In [57]:
```
pred
```

Out[57]:
```
array([[0., 0., 0., 1.]], dtype=float32)
```

In [58]:
```
prediction_class = class_names[np.argmax(pred)]
prediction_class
```

Out[58]: 'Tungro'

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: