In [2]:
```
import tensorflow as tf
from tensorflow.keras import models,layers
```

here we will load the data into tensor flow

In [4]:
```
print(tf.config.list_physical_devices('GPU'))
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

In [7]:
```
IMAGE_SIZE = 256
BATCH_SIZE = 16
CHANNELS = 3
EPOCHS = 50
```

In [9]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage/Plant",
    shuffle = True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
```

```
Found 13692 files belonging to 16 classes.
```

In [11]:
```
def get_partition_and_data_tuning(ds,train_split= 0.8 ,val_split=0.1,test_split = 0
    train_size = int(len(dataset)*train_split)
    val_size = int(len(dataset)*val_split)
    test_size = int(len(dataset)*test_split)
    if shuffle:
        ds = ds.shuffle(shuffle_size,seed = 12)
    train_data = ds.take(train_size)
    remain_data = ds.skip(train_size)
    val_data = remain_data.take(val_size)
    test_data = remain_data.skip(val_size)
    train_data = train_data.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTO
    val_data = val_data.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE
    test_data = test_data.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTU
    return train_data,val_data,test_data
```

In [13]:
```
train,val,test = get_partition_and_data_tuning(dataset)
```

In [15]:
```
resize_and_rescale = tf.keras.Sequential([

    layers.experimental.preprocessing.Resizing(IMAGE_SIZE,IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255),
])
```

In [17]:
```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
])
```

In [19]:
```
input=(IMAGE_SIZE,IMAGE_SIZE,CHANNELS)
n_classes = 11
```

In [21]:
```
model = tf.keras.Sequential([
    layers.Conv2D(32,(3,3),activation="relu",input_shape=input),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,(3,3),activation="relu"),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64,activation="relu"),
    layers.Dense(n_classes,activation="softmax")
])
```

In [23]:
```
model.build(input_shape = input)
```

In [25]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 28, 28, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 12, 12, 64)        36928

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_5 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dense_1 (Dense)             (None, 11)                715

=================================================================
Total params: 184,267
Trainable params: 184,267
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```python
with tf.device('/device:GPU:0'):
    model.compile(
        optimizer='adam',
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
        metrics=['accuracy']
    )
    history = model.fit(
        train,
        epochs = EPOCHS,
        batch_size = BATCH_SIZE,
```

```
        verbose = 1,
        validation_data = val
    )
```

```
Epoch 1/50
271/684 [==========>...................] - ETA: 33s - loss: nan - accuracy: 0.0473
```

In [15]:
```
scores = model.evaluate(test)
```

```
61/61 [==============================] - 25s 53ms/step - loss: 0.1069 - accuracy: 0.
9688
```

In [16]:
```
scores
```

Out[16]:
```
[0.10689208656549454, 0.9688149690628052]
```

In [17]:
```
history.params
```

Out[17]:
```
{'verbose': 1, 'epochs': 50, 'steps': 482}
```

In [18]:
```
history.history.keys()
```

Out[18]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [19]:
```
len(history.history['accuracy'])
```

Out[19]:
```
50
```

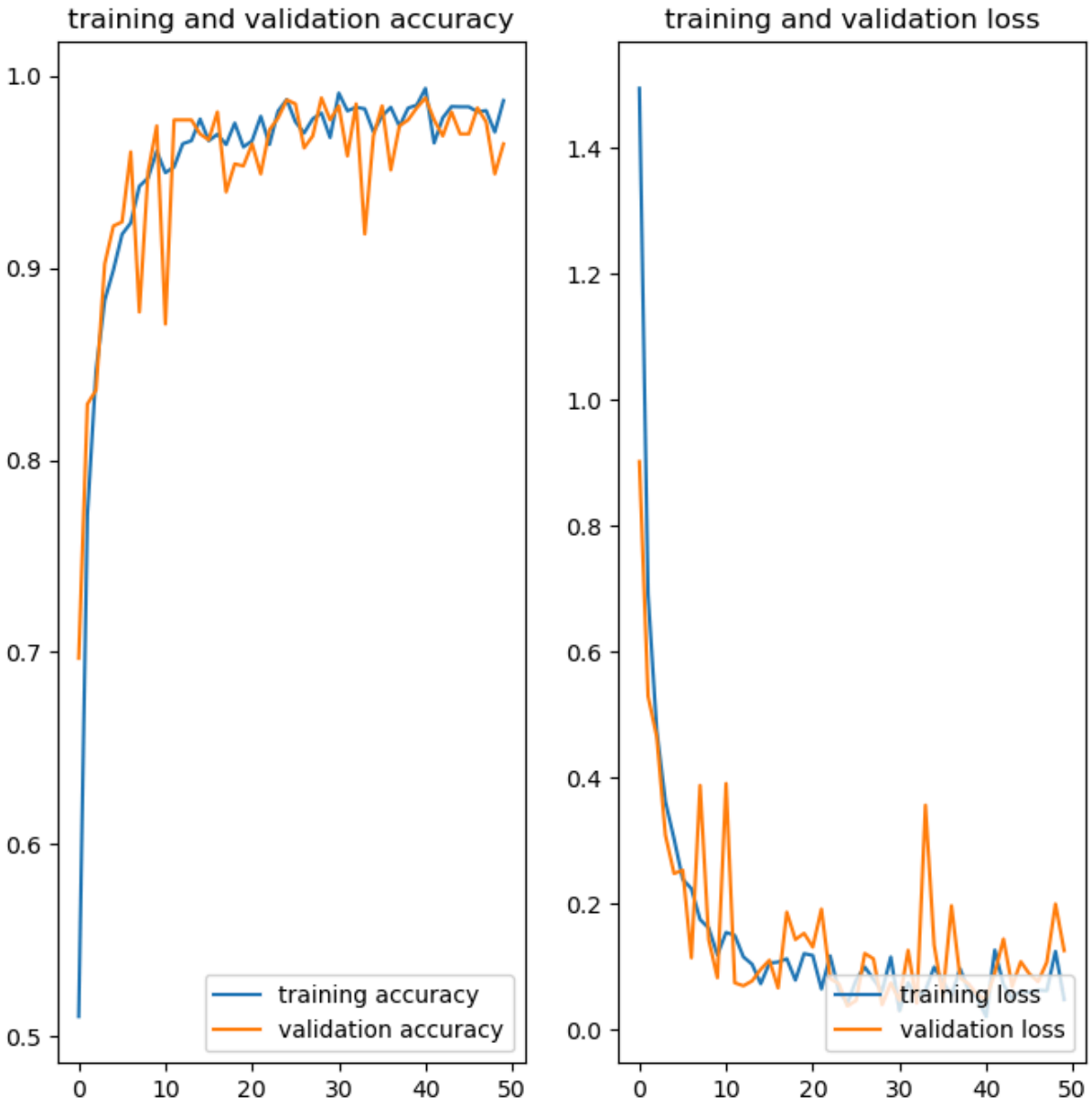
In [20]:
```
import matplotlib as mlt
```

In [21]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [22]:
```
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")

plt.subplot(1,2,2)
plt.plot(range(EPOCHS),loss,label="training loss")
plt.plot(range(EPOCHS),val_loss,label="validation loss")
plt.legend(loc="lower right")
plt.title("training and validation loss")
```

Out[22]:
```
Text(0.5, 1.0, 'training and validation loss')
```



In [23]:
```
import os
os.listdir("../leaf disease prediction/Models")
```

Out[23]:
```
['.ipynb_checkpoints',
 '2',
 'apple',
 'apple_aware_model',
 'cherry',
 'cherry_aware_model',
 'peach',
 'peach_aware_model',
 'pepper',
 'pepper_aware_model',
 'potato',
 'potato_aware_model',
 'quantize_aware_model',
 'straberry',
 'straberry_aware_model']
```

In [24]:
```
# version = max([int(i) for i in  os.listdir("../leaf disease prediction/Models")+[
model.save(f"../leaf disease prediction/Models/quantize")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_com
piled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _j
it_compiled_convolution_op while saving (showing 5 of 6). These functions will not b
e directly callable after loading.
INFO:tensorflow:Assets written to: ../leaf disease prediction/Models/quantize\assets
INFO:tensorflow:Assets written to: ../leaf disease prediction/Models/quantize\assets
```

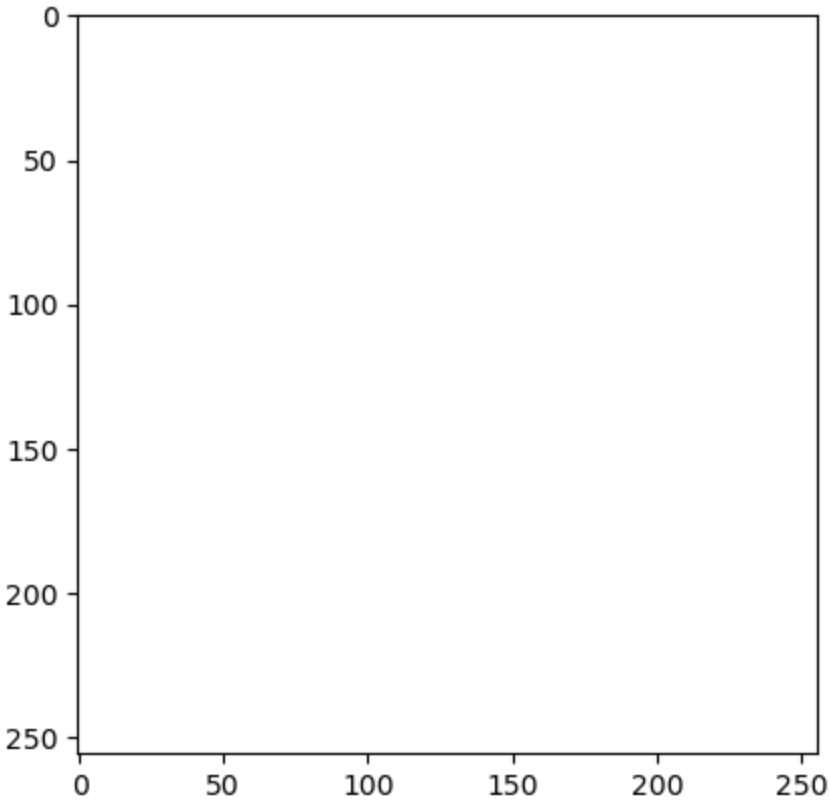
In [ ]:
```
```

In [25]:
```
# _____predicting_____
```

In [26]:
```
import numpy as np
```

In [27]:
```
for images_batch,labels_batch in test.take(1):
    first_img = images_batch[0].numpy().astype('float32')
    first_label = labels_batch[0].numpy()
    print("first image to predict")
    plt.imshow(first_img)
    print("actual label:",dataset.class_names[first_label])
    batch_prediction = model.predict(images_batch)
    label1 = np.argmax(batch_prediction[0])
    print("predicted label:",dataset.class_names[label1])
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB
data ([0..1] for floats or [0..255] for integers).
first image to predict
actual label: Apple__Cedar_apple_rust
1/1 [==============================] - 0s 301ms/step
predicted label: Apple__Cedar_apple_rust
```



In [28]:
```
def predict(model,img):

    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array,0)#create a batch
    prediction = model.predict(img_array)
    pre_class = dataset.class_names[np.argmax(prediction[0])]
    confidence = round(100*(np.max(prediction[0])),2)
    return pre_class,confidence
```

In [29]:
```
#_____________________________________loding and using the saved model_____________
```

In [30]:
```
from tensorflow.keras.models import load_model
model = load_model("../leaf disease prediction/Models/quantize")
```

In [ ]:
```

```

In [31]:
```
import tensorflow_model_optimization as tfmot
```

In [32]:
```
new_model = tf.keras.models.load_model('Models/quantize')
```

In [33]:
```
quantize_model = tfmot.quantization.keras.quantize_model
```

In [34]:
```
quantize_aware_model = quantize_model(new_model)
```

In [35]:
```
quantize_aware_model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy'])
```

In [36]:
```
quantize_aware_model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
quantize_layer (QuantizeLayer)	(None, 256, 256, 3)	3
quant_conv2d (QuantizeWrapperV2)	(None, 254, 254, 32)	963
quant_max_pooling2d (QuantizeWrapperV2)	(None, 127, 127, 32)	1
quant_conv2d_1 (QuantizeWrapperV2)	(None, 125, 125, 64)	18627
quant_max_pooling2d_1 (QuantizeWrapperV2)	(None, 62, 62, 64)	1
quant_conv2d_2 (QuantizeWrapperV2)	(None, 60, 60, 64)	37059
quant_max_pooling2d_2 (QuantizeWrapperV2)	(None, 30, 30, 64)	1
quant_conv2d_3 (QuantizeWrapperV2)	(None, 28, 28, 64)	37059
quant_max_pooling2d_3 (QuantizeWrapperV2)	(None, 14, 14, 64)	1
quant_conv2d_4 (QuantizeWrapperV2)	(None, 12, 12, 64)	37059
quant_max_pooling2d_4 (QuantizeWrapperV2)	(None, 6, 6, 64)	1
quant_conv2d_5 (QuantizeWrapperV2)	(None, 4, 4, 64)	37059
quant_max_pooling2d_5 (QuantizeWrapperV2)	(None, 2, 2, 64)	1
quant_flatten (QuantizeWrapperV2)	(None, 256)	1
quant_dense (QuantizeWrapperV2)	(None, 64)	16453
quant_dense_1 (QuantizeWrapperV2)	(None, 11)	720

Total params: 185,009
Trainable params: 184,267

```
Non-trainable params: 742
_________________________________________________________________
```

In [37]:
```
with tf.device('/device:GPU:0'):
    quantize_aware_model.fit(
        train,
        epochs = 10,
        batch_size = BATCH_SIZE,
        verbose = 1,
        validation_data = val
    )
```

```
Epoch 1/10
482/482 [==============================] - 108s 219ms/step - loss: 0.2550 - accurac
y: 0.9224 - val_loss: 0.0998 - val_accuracy: 0.9719
Epoch 2/10
482/482 [==============================] - 77s 159ms/step - loss: 0.0975 - accuracy:
0.9729 - val_loss: 0.2510 - val_accuracy: 0.9375
Epoch 3/10
482/482 [==============================] - 69s 143ms/step - loss: 0.1286 - accuracy:
0.9725 - val_loss: 0.2399 - val_accuracy: 0.9521
Epoch 4/10
482/482 [==============================] - 75s 154ms/step - loss: 0.0825 - accuracy:
0.9808 - val_loss: 0.0680 - val_accuracy: 0.9854
Epoch 5/10
482/482 [==============================] - 68s 142ms/step - loss: 0.0583 - accuracy:
0.9834 - val_loss: 0.1412 - val_accuracy: 0.9688
Epoch 6/10
482/482 [==============================] - 74s 154ms/step - loss: 0.0877 - accuracy:
0.9799 - val_loss: 0.1356 - val_accuracy: 0.9667
Epoch 7/10
482/482 [==============================] - 79s 163ms/step - loss: 0.0495 - accuracy:
0.9856 - val_loss: 0.1833 - val_accuracy: 0.9490
Epoch 8/10
482/482 [==============================] - 73s 151ms/step - loss: 0.0581 - accuracy:
0.9852 - val_loss: 0.1010 - val_accuracy: 0.9802
Epoch 9/10
482/482 [==============================] - 74s 154ms/step - loss: 0.0562 - accuracy:
0.9845 - val_loss: 0.0740 - val_accuracy: 0.9802
Epoch 10/10
482/482 [==============================] - 65s 134ms/step - loss: 0.0235 - accuracy:
0.9923 - val_loss: 0.0704 - val_accuracy: 0.9844
```

In [38]:
```
###### quantize_aware_model.evaluate(test)
quantize_aware_model.save(f"../leaf disease prediction/Models/quantize_aware_model"
```

```
WARNING:absl:Found untraced functions such as conv2d_layer_call_fn, conv2d_layer_cal
l_and_return_conditional_losses, _jit_compiled_convolution_op, conv2d_1_layer_call_f
n, conv2d_1_layer_call_and_return_conditional_losses while saving (showing 5 of 24).
These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: ../leaf disease prediction/Models/quantize_aware_
model\assets
INFO:tensorflow:Assets written to: ../leaf disease prediction/Models/quantize_aware_
model\assets
```

In [39]:
```
convertor = tf.lite.TFLiteConverter.from_keras_model(quantize_aware_model)
convertor.optimizations = [tf.lite.Optimize.DEFAULT]
tf_model1 = convertor.convert()
```

```
WARNING:absl:Found untraced functions such as conv2d_layer_call_fn, conv2d_layer_cal
l_and_return_conditional_losses, _jit_compiled_convolution_op, conv2d_1_layer_call_f
n, conv2d_1_layer_call_and_return_conditional_losses while saving (showing 5 of 24).
These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: C:\Users\heman\AppData\Local\Temp\tmpbpmpwed5\ass
ets
```

```
INFO:tensorflow:Assets written to: C:\Users\heman\AppData\Local\Temp\tmpbpmpwed5\ass
ets
C:\Users\heman\anaconda3\lib\site-packages\tensorflow\lite\python\convert.py:766: Us
erWarning: Statistics for quantized inputs were expected, but not specified; continu
ing anyway.
  warnings.warn("Statistics for quantized inputs were expected, but not "
```

In [40]:
```
with open("quantize_tflite_model.tflite","wb") as f:
    f.write(tf_model1)
```

In [3]:
```
nbconvert --to webpdf --allow-chromium-download modeltraining.ipynb
```

```
  Cell In[3], line 1
    nbconvert --to webpdf --allow-chromium-download modeltraining.ipynb
                  ^
SyntaxError: invalid syntax
```

In [ ]: